



UNIVERSITY OF PADUA

PH.D. SCHOOL IN BRAIN, MIND AND COMPUTER SCIENCE

CURRICULUM IN COMPUTER SCIENCE FOR SOCIETAL CHALLENGES AND INNOVATION

XXXVI CYCLE

SECURE IoT AND THE IDENTITY OF THINGS

SUPERVISOR

PROF. MAURO CONTI
UNIVERSITY OF PADUA

CO-SUPERVISOR

PROF. ANNA SPAGNOLLI
UNIVERSITY OF PADUA

CANDIDATE

ALESSANDRO VISINTIN

THESIS WRITTEN WITH THE FINANCIAL CONTRIBUTION OF INFOCERT S.P.A

“THE SHIP WHEREIN THESEUS AND THE YOUTH OF ATHENS RETURNED FROM CRETE HAD THIRTY OARS, AND WAS PRESERVED BY THE ATHENIANS DOWN EVEN TO THE TIME OF DEMETRIUS PHALEREUS, FOR THEY TOOK AWAY THE OLD PLANKS AS THEY DECAYED, PUTTING IN NEW AND STRONGER TIMBER IN THEIR PLACES, INSOMUCH THAT THIS SHIP BECAME A STANDING EXAMPLE AMONG THE PHILOSOPHERS, FOR THE LOGICAL QUESTION OF THINGS THAT GROW; ONE SIDE HOLDING THAT THE SHIP REMAINED THE SAME, AND THE OTHER CONTENDING THAT IT WAS NOT THE SAME.”

— PLUTARCH, LIFE OF THESEUS 23.1

Abstract

The rapid proliferation of digital technologies and the exponential growth of the Internet of Things have made digital identities the cornerstone of the current digital revolution. As our world becomes increasingly interconnected, the need for secure digital means of authentication has become paramount. However, this presents a significant challenge, particularly for small devices within the Internet of Things ecosystem.

This thesis explores three stages in the evolution in digital identities. The first part investigates traditional digital identities, characterized by the use of cryptographic secrets for authentication. Addressing secure storage concerns, researchers have harnessed physical unclonable functions to fortify these identities. The second part explores more sophisticated identity representations, encompassing a group of properties. Verifiable credentials and selective disclosure mechanisms are examined to ensure the privacy and integrity of data. The third part delves into the realm of artificial intelligence as a potential means for flexible identity representation. Large language models emerge as powerful tools for dynamically assembling groups of properties into coherent identities. Combining these AI-enhanced representations into verifiable credentials forms the foundation for a flexible identification framework.

Contents

ABSTRACT	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
LISTING OF ACRONYMS	xiii
1 INTRODUCTION	1
1.1 Research motivation and contributions	2
1.2 Understanding the Internet of Things	5
1.3 Digital identity and the Identity of things	7
1.4 Modern solutions to an old problem: the ship of Theseus tale	8
1.5 What You See is <i>Not</i> What You Get: a Man-in-the-Middle Attack Applied to Video Channels	9
1.5.1 System and threat model	10
1.5.2 Man-in-the-Video: a MITM on video channels	11
1.5.3 HackDMI: a MITV on HDMI channels	13
1.5.4 Analysis and results	18
1.5.5 Related works	20
1.5.6 Discussion	21
2 TRADITIONAL SINGLE-ATTRIBUTE IDENTIFICATION	23
2.1 HolA: Holistic and Autonomous Attestation for IoT Networks	24
2.1.1 Background	25
2.1.2 Assumptions	26
2.1.3 Motivation	27
2.1.4 HolA Overview	28
2.1.5 HolA Design	30
2.1.6 HolA Evaluation	32
2.1.7 Related Works	35
2.1.8 Discussion	36
2.2 X-Lock: a secure XOR-based fuzzy extractor for resource-constrained devices	38
2.2.1 Related Works	39
2.2.2 Background	40
2.2.3 X-Lock: construction details	44
2.2.4 X-Lock: algorithm analysis	45
2.2.5 Costs analysis	51
2.2.6 Implementation and comparison	52
2.2.7 Discussion	55
3 EMBRACING COMPLEXITY WITH MULTI-ATTRIBUTE IDENTITIES	57

3.1	Location-wise anomaly detection on Twitter: unveiling followers anomalies during the 2020 US Election campaign	58
3.1.1	Related works	59
3.1.2	Background	60
3.1.3	Building a location-wise classifier	61
3.1.4	Reverse engineering Twitter timestamped lists	66
3.1.5	Results analysis	68
3.1.6	Discussion	72
3.2	Cryptographic methods for Selective Disclosure in Verifiable Credentials	73
3.2.1	Self-Sovereign Identity	73
3.2.2	Verifiable Credentials	75
3.2.3	Cryptographic preliminaries	76
3.2.4	Protocols for Zero-Knowledge Proof of Knowledge	80
3.2.5	Selective disclosure schemes	82
4	TOWARD A DYNAMIC IDENTITY FRAMEWORK FOR IoT	89
4.1	Technologies for representing knowledge	90
4.2	The problem of conceptual organization: the semantic Web case	91
4.3	Large language models for flexible identities	93
4.4	ChatGPT's Role in Cybersecurity: A Critical Examination of Code Vulnerability Detection	95
4.4.1	Related works	97
4.4.2	Motivation	98
4.4.3	Methodology	100
4.4.4	Results	101
4.4.5	Discussion	102
4.5	Conclusion	103
	REFERENCES	105

Listing of figures

1.1	Proposed IoT architectures.	6
1.2	Graphical representation of the system model.	11
1.3	Graphical representation of the attacker placed in the middle of the video channel.	12
1.4	Configuration for a Man-in-the-Video attack.	12
1.5	Minimal implementation for the hardware involved in a Man-in-the-Video attack.	13
1.6	Software pipeline for HackDMI.	15
1.7	FPS generated at different stages of the elaboration process (SS - SimpleStream, BB - Bounding-Boxes, FD - FilterDimensions, RS - RowSeparation, SW - SeparateWords, FL - FilterLength, DP - DetectPattern, MI - ModifyImage). The graphs plot the averages of 5 different experiments. The shaded area represents the standard deviation.	19
2.1	Simplified Chord organization. Each node is connected to its direct <i>successors</i> and contains a <i>finger table</i> to speed up seeking operation.	26
2.2	Lifecycle of a network node deployed in the HoLA CRA scheme.	29
2.3	Minimum successors list length for preserving the ring structure (a), average time between two consecutive challenges (b), average time required for a broadcast (c), average number of messages for broadcasting (d), maximum fingers list size (e), average number of devices communicating with a node (f). The lines represent offline nodes (a,b) and compromised nodes (c,d). Shaded areas are the standard deviation.	37
2.4	Composition of the vault V . It is a collection of bit-lockers, each referring to a specific bit $b \in B$. A bit-locker contains multiple locks that XOR the bit of reference with the XOR-ation of a random subset of bits from the source F	44
2.5	Impact of correlation on bit security with varying number of elements in the XOR cascade. The x-axis considers the number of defined bits from the source. The y-axis shows the corresponding number of exposed bits from the random pool. Using a single element provides virtually no security. From three elements onwards the number of defined bits is greater than those in the protected pool (vertical, dashed line).	51
2.6	Number of locks required to achieve a given error tolerance for varying errors. The x-axis considers the error. The y-axis uses logarithmic scale and considers the number of lockers. The gray, dashed lines provides a linear reference for the four curves. The curves show an initial linear trend that slowly diverges to exponential at around value 0.85 for the error.	53
2.7	XOR-ation size while combining multiple locks. The x-axis considers the number of locks added. The y-axis considers the number of elements forming the XOR-ation. Both axes are logarithmic. The graph considers the worst case, that is the minimum number of elements in the resulting cascade.	55
3.1	Embedding for the Latin alphabet projected to 2D space using t-SNE. The vowels are grouped together, showing that the model captured more complex structures.	64
3.2	Cursors act as time pointers in the followers timeline. When a cursor retrieves a batch of IDs, the system returns a list starting from the ID that temporally follows the cursor. Decrementing the cursor does not change the result, while incrementing it returns a list with the last ID of the current batch on top.	69

3.3	Followers trend of @JoeBiden divided by ethnicity during 2020. The thinner, gray line represents the followers trend in logarithmic scale. The solid black line illustrates the percentage trend of Sub-Indian profiles. The dotted red line represents the percentage trend of Spanish profiles. The dashed green line portrays the percentage trend of French profiles. Lastly, the dash-dotted blue line displays the percentage trend of Arabic profiles. The anomalous trends detectable before the announcement of Kamala Harris as vice-president supports the rumors of an usage of Sub-Indian fake followers and suggest an additional usage of Arabic fake followers. Another anomaly emerges for the Sub-Indian group approximately two weeks before the Election day, further supporting the presence of fraudulent activities.	70
3.4	Kernel density regions of 4 class A continuous features from [1]. The data refers to the period between July 21st, 2020 and August 21st, 2020. Graph (a) depicts the logarithm of the ratio between the number of friends and the number of followers squared. Graph (b) shows the age of the accounts. Graph (c) displays the logarithm of the number of tweets. We represent 0 values with -1 . Graph (d) depicts the logarithm of the number of friends. The 4 lines show the trend of the locations of reference.	71
3.5	General architecture for a Verifiable Credentials system.	76

Listing of tables

2.1	Main components of the HolA devices.	30
2.2	Notation summary.	43
2.3	Parameters for the three security levels.	53
2.4	Performance comparisons	54
3.1	Class A features as proposed in Cresci et al. [1]. In bold the features we considered in our analysis. The column acronym summarizes the shortcut we use throughout the section to reference the specific feature.	62
3.2	Twitter profiles used for creating the location-wise dataset. The countries were selected based on the associated user base on Twitter. Each profile was chosen for its specificity to the respective country. We focused on profiles with moderate-sized follower bases to reduce the risk of encountering numerous fake accounts.	63
3.3	Clusters of countries used to train the second model. The clustering was suggested by the results of the first model and are further supported by cultural similarities.	65
3.4	Confusion matrix for the description model.	66
3.5	Confusion matrix for the location model.	67
3.6	Confusion matrix for the name model.	68
3.7	Class A boolean features from [1]. The numbers represent the fraction of cluster population that satisfies (does not satisfy) the relative feature. The data refers to the period between July 21st, 2020 and August 21st, 2020.	72
3.8	Schnorr's Identification scheme. g, q, y are the public parameters. x is the prover's secret . . .	81
3.9	Camenisch-Lysyanskaya signature scheme. k denotes the security parameter. p, q are safe primes. QR is the group of quadratic residues. M denotes the set of messages $\{(m_0, \dots, m_{L-1}) : m_i \in \pm\{0, 1\}^{l_m}\}$. All operations are $\text{mod } n$	83
3.10	Camenisch-Lysyanskaya Selective Disclosure scheme. k denotes the security parameter. A, S, Z are calculated as in protocol 3.9. $G = \langle g_0 \rangle$ denotes a group of prime order $q > 2^{l_m}$. $g_1, \dots, g_L$ are elements of G . All operations are $\text{mod } n$	84
3.11	Boneh-Boyen-Shacham signature scheme.	85
3.12	Boneh-Boyen-Shacham selective disclosure scheme.	86
3.13	Zero-knowledge proof of knowledge of a single polynomial. PK denotes the proving key, while VK the verification key.	88

Listing of acronyms

2FA	Two-factor authentication
AI	Artificial Intelligence
API	Application Programming Interface
APR	Automatic program repair
BBS	Boneh-Boyen-Shacham
BERT	Bidirectional Encoder Representations from Transformers
CA	Certification Authority
CDH	Computational Diffie-Hellman
CL	Camenisch-Lysyanskaya
CRA	Collective Remote Attestation
CRP	Challenge-Response Pair
CWE	Common Weakness Enumeration
DDH	Decisional Diffie-Hellman
DHT	Distributed Hash Table
DID	Decentralized Identifier
DoS	Denial of Service
EDID	Extended Display Identification Data
FPS	Frames-Per-Second
GPT	Generative Pretrained Transformer
HDMI	High-Definition Multimedia Interface
IdM	Identity Management
IDOT	Identity of Things
IoT	Internet of Things
JSON	JavaScript Object Notation
KG	Knowledge Graph
KR	Knowledge Representation
LLM	Large Language Model
LWE	Learning With Errors
MITM	Man-in-the-Middle

MITV Man-in-the-Video
NLP Natural Language Processing
NVM Non-Volatile Memory
OWL Ontology Web Language
PKI Public Key Infrastructure
PUF Physical Unclonable Function
q-sDH q-strong Diffie-Hellman
RA Remote Attestation
RBA Risk-Based Authentication
RDF Resource Description Framework
RSA Rivest-Shamir-Adleman
SAN Semantic Actuator Network
SD Selective Disclosure
SL Status List
SoC System-on-Chip
SRAM Static Random Access Memory
SSI Self-Sovereign Identity
SSN Semantic Sensor Network
UML Unified Modeling Language
VC Verifiable Credential
VGA Video Graphics Array
W₃C World Wide Web Consortium
WiFi Wireless Fidelity
XML eXtensible Markup Language
ZKP Zero-Knowledge Proof
zk-SNARK Zero-Knowledge Succint Non-interactive Arguments of Knowledge

1

Introduction

1.1 RESEARCH MOTIVATION AND CONTRIBUTIONS

The **IoT** is on the rise and promises to revolutionize the digital world. Billions of low-end devices are currently connected to the Internet, exchanging data and remotely controlling actuators. While protecting these devices from malicious tampering is of primary importance, their limited capabilities and particular boundary conditions render the task challenging. Attackers have already exploited the lowered level of security, causing severe damages to the Internet infrastructure. The Mirai botnet [2] is one notable example of a successful attack that leveraged on poor security standards. Ransomware campaigns are another example of large attacks that undermine private and public organizations. The numbers of this type of attack is exponentially increasing, passing from 304 millions in 2020 to 623 millions in 2021 [3]. In this context, digital identities are prime security components used for protecting **IoT** appliances. However, directly porting consolidated techniques to the restricted **IoT** environment has proven to be a difficult objective. This thesis investigates the situation by following three intertwined paths: 1. *Single-attribute identities*. Traditional identities rely on cryptographic secrets to provide a mean of identification. Chapter 2 introduces HolA, an autonomous attestation system based on single-attribute authentication, and X-Lock, a secure fuzzy extractor for restoring cryptographic secrets from a physical unclonable function. 2. *Multi-attribute identities*. The composite nature of the physical world demands sophisticated identities formed by groups of attributes. Chapter 3 explores ethnicity identification on Twitter through groups of attributes, and surveys cryptographic solutions to implement selective disclosure for multi-attribute identities. 3. *Dynamic identities*. Attributes for identification are subjective and thus hard to determine. Chapter 4 assesses the capabilities of large language models to model knowledge and proposes a potential scheme of a framework for flexible representation of identities. As a support to the overall exposition, the thesis presents some of the works that have been produced during the Ph.D. period.

What You See is *Not* What You Get: a Man-in-the-Middle Attack Applied to Video Channels. People usually think that digital screens are reliable devices. Unfortunately, attackers can exploit this blind trust to persuade a user to perform unintended actions. Section 1.5 presents a novel type of **MITM** attack named **MITV**. **MITV** intercepts the video stream flowing between a computer and its screen and modifies it on-the-fly. The objective of such attack is to distort the perception of reality and to induce improper user behaviour. We implemented HackDMI, a **MITV** attack performed over an **HDMI** cable. We applied this attack to a realistic threat scenario (*i.e.*, phishing) and we evaluated it with quantitative measures. HackDMI is able to deceptively modify a 720p video stream, while maintaining a frame-rate of 14FPS. We also recorded three demo videos for qualitative evaluation.

HolA: Holistic and Autonomous Attestation for IoT Networks. **CRA** is a well-established approach where a single *Verifier* attests the integrity of multiple devices in a single execution of the challenge-response protocol. Current **CRA** solutions are well-suited for **IoT** networks, where the devices are distributed in a mesh topology and communicate only with their physical neighbours. Recent advancements on low-energy protocols, though, enabled the **IoT** devices to be connected to the Internet, thus disrupting the concept of physical neighbour. Section 2.1 proposes HolA (Holistic and Autonomous Attestation), the first **CRA** scheme designed for Internet-like **IoT** networks. HolA provides defence against attacks targeting both the nodes and the network infrastructure. We deployed HolA on both a network of real devices (*i.e.*, 5 Raspberry Pis) and a simulated environment (*i.e.*, 1M devices in an Omnet++ network). Our results demonstrate that HolA can resist against a disruptive attacker that compromises up to half of the network devices and that tampers with network traffic. HolA can verify the

integrity of 1M devices in around 12 seconds while the state-of-the-art requires 71 seconds. Finally, HolA requires 7 times less memory per device compared with the state-of-the-art.

X-Lock: a secure XOR-based fuzzy extractor for resource constrained devices. The rapid expansion of resource-constrained devices in the **IoT** domain has caused considerable challenges concerning privacy and security. Traditional methods of storing digital keys in non-volatile memories have proven intricate and financially burdensome. As an alternative, silicon’s **PUFs** have emerged as a promising solution, capitalizing on inherent process variations to extract confidential information and effectively thwart replication attempts. However, the innate fuzziness of physical unclonable functions necessitates the adoption of fuzzy extractors to ensure reliable replication of responses. Regrettably, current approaches encounter issues like sensitive information leakage and high computational and memory requirements. Section 2.2 introduces X-Lock, a novel and secure computational fuzzy extractor that addresses the limitations faced by traditional solutions in resource-constrained **IoT** devices. X-Lock offers a reusable and robust solution, effectively mitigating the impacts of bias and correlation through its design. Leveraging the preferred state of a noisy source, X-Lock encrypts a random string of bits that can be later used as seed to generate multiple secret keys. To prove our claims, we provide a comprehensive theoretical analysis, addressing both security considerations and implementation details of the proposed model. To evaluate the effectiveness and superiority of our approach, we also propose practical experiments and compare the results with existing approaches. The experimental findings demonstrate the efficacy of our proposed algorithm, showing that the model has comparable memory cost ($\approx 5KB$ for storing 9 keys of 128 bits) while being 3 orders of magnitude faster with respect to the state-of-the-art solutions (1.32ms against 7.0s).

Location-wise anomaly detection on Twitter: unveiling followers anomalies during the 2020 US Election campaign. Twitter has emerged as a prominent political arena, where public figures strategically utilize the platform to amass visibility and garner public consent. However, this digital landscape is not immune to illicit behaviors, with the marketing of fake followers posing a significant concern for the transparency of digital discourse. Section 3.1.3 investigates the potential of location-wise analysis to identify anomalies within the followers of political figures. We developed a neural network to effectively infer the geographical location of users based on features available in their profiles. We then applied the model to analyze the timeline of followers associated with Joe Biden. Our findings reveal conspicuous anomalies in the trends observed during the 2020 US Election campaign, specifically among Sub-Indian (India and Pakistan) and Arabic (Egypt and Saudi Arabia) profiles. Further analyses on the features confirm the anomalous nature of the involved users, thus providing support to the claims regarding the usage of fake followers associated with Joe Biden’s profile.

Cryptographic methods for Selective Disclosure of Verifiable Credentials. Section 3.2 explores the critical aspects of **VCs** and **SD** protocols, exploring their importance in modern digital identity systems. **VCs** allow individuals to assert and demonstrate their credentials without disclosing unnecessary personal information, promoting privacy and security in various applications. The study provides a comprehensive analysis of three prominent methodologies to implement **VCs** and **SD**: Camenisch-Lysyanskaya, Boneh-Boyen-Schacam and **zk-SNARKs**. Camenisch-Lysyanskaya’s methodology is examined for its efficiency in allowing selective disclosure through the use of non-transferable anonymous credentials, ensuring privacy while enabling credential validation. Boneh-Boyen-Schacam’s approach is explored for its ability to facilitate selective disclosure through the aggregation of credentials, reducing the computational burden in verification processes. In addition, the article investigates the applicability and effectiveness of **zk-SNARKs** in implementing **VCs** and **SD**, emphasizing their role in achieving

concise and privacy-preserving evidence.

ChatGPT's Role in Cybersecurity: A Critical Examination of Code Vulnerability Detection. Recent advances in generative AI methodologies have had a substantial impact in the field of software engineering. AI-based approaches have played a crucial role in addressing common challenges faced by developers, such as correcting faulty programs. A notable development in this area is the increased adoption of LLMs, as demonstrated by OpenAI's Codex. Among these models, ChatGPT, a prominent LLM, attracted considerable attention. It serves as a versatile tool for conducting discussions about source code, offering suggestions for changes, providing explanations and even generating snippets of code. These demonstrations usually focus on established benchmarks, which may have been used in model formation (raising concerns about data loss). In order to assess the practicality of using an LLM as an effective programming assistant, it becomes imperative to analyze its capabilities on previously unresolved challenges. In section 4.4, we present an empirical examination of the potential of ChatGPT as an automated programming assistant, with a specific focus on its performance in vulnerability detection tasks. The study examines the effectiveness of ChatGPT using the widely recognized Juliet dataset to assess its realistic performance in detecting vulnerabilities. To achieve this, we used three levels of code refactoring and used two distinct prompting techniques. The results showed a decrease in performance when the code is refactored from 54% to 34%. This decrease supports the possibility of dataset inclusion during the training of the model. Moreover, the adoption of a more sophisticated prompting technique seems to produce more accurate responses, albeit with some limitations. Our findings provide valuable insights into LLM's development in programming assistance, putting particular emphasis on the importance of prompting engineering and offering a deeper understanding of the practical utility of ChatGPT in the field of software engineering.

In addition, the Ph.D. period brought to the completion of several works that have been either published or submitted to peer-reviewed journals and conferences. The following list reports them in chronological order by acceptance and submission date.

Journal publications.

1. Alessandro Visintin, Alessandro Brighente, Mauro Conti - Cryptographic methods for selective disclosure in Verifiable Credentials - SUBMITTED *Computers & Security (COSE)*

Conference publications.

1. Mauro Conti, Eleonora Losiouk, Alessandro Visintin - 2022 - What You See is Not What You Get: a Man-in-the-Middle Attack Applied to Video Channels - *Computer security track at the 37th ACM/SIGAPP Symposium On Applied Computing (SEC@SAC2022)*
2. Alessandro Visintin, Flavio Toffalini, Eleonora Losiouk, Mauro Conti, Jianying Zhou - 2021 - HolA: Holistic and Autonomous Attestation for IoT Networks - *4th International Workshop on Artificial Intelligence and Industrial Internet-of-Things Security (AIIoTS@ACNS 2022)*
3. Alessandro Visintin, Mauro Conti - Leveraging social networks for mergers and acquisitions forecasting - *24th International Conference on Web Information Systems Engineering (WISE2023)*
4. Alessandro Visintin, Edoardo Liberati, Mauro Conti, Riccardo Lazzeretti - X-Lock: a secure XOR-based fuzzy extractor for resource constrained devices - SUBMITTED *22nd International Conference on Applied Cryptography and Network Security (ACNS2024)*

5. Alessandro Visintin, Mauro Conti - Location-wise anomaly detection on Twitter: unveiling followers anomalies during the 2020 US Election campaign - SUBMITTED *9th IEEE European Symposium on Security and Privacy (EuroSecP2024)*

Other results.

1. PATENT GRANTED n. 102021000007808 - Dispositivo e sistema per la trasmissione protetta di un segnale video.
2. PATENT GRANTED n. 102021000001022 - Dispositivo atto ad essere connesso tra un computer ed uno schermo per modifica in tempo reale di un segnale video digitale in uscita da detto computer verso detto schermo.
3. FUNDED PROJECT Revolutionised Enhanced Supply Chain Automation with Limited Threats Exposure (RESCALE) – Horizon Europe Framework Programme (HORIZON-CL3-2022-CS-01-02).

1.2 UNDERSTANDING THE INTERNET OF THINGS

As technology advances, global connectivity across the Internet is increasing [4]. The **IoT** has emerged as a prominent technology, efficiently connecting billions of devices such as sensors, actuators, gateways and controllers, offering numerous applications in various industries [5]. The **IoT** connects the physical and virtual world through different mediums, finding applications in fields such as engineering, medicine, finance, food, energy and agriculture. Novel **IoT** devices, operating systems, architectures, platforms, security measures, and communication protocols continue to emerge, leading to developments that continually expand its shape. **IoT** applications can be broadly classified into three categories [6]: consumer-based (*e.g.*, home automation, autonomous vehicles, smart wearables, automated healthcare, smart homes), industry-based (*e.g.*, smart manufacturing, retail, industrial automation), and infrastructure-based (*e.g.*, smart cities, habitat monitoring, environmental monitoring, smart grids). The **IoT** ecosystem efficiently integrates various components, including devices, operating systems, controllers, gateways, middleware, and platforms. These elements connect through communication protocols [7] (*e.g.*, Zigbee, low-power **WiFi**, MQTT, 6LoWPAN, NFC, BLE), creating an environment that connects a multitude of physical devices. By 2025, it is estimated that this ecosystem will include 19 billion devices [8]. Several researchers propose different **IoT** architectures. The usual architecture comprises three levels [9]: perception (management of sensors and implementation), network (management of information transmission and processing) and application (provision of user-specific applications). Expanding on this concept, a five-level architecture adds abstraction, including levels of perception, transport, processing, middleware, and application [10]. The transport layer moves the sensor data to the processing level, where it is analysed [11]. In middleware architecture, the middleware level is crucial to manage and control the data flow in the system. The physical plane includes levels of perception, access and edge with sensors and actuators. The virtualized plane has the Backbone Network Level and the Middleware level with cloud and servers. The Coordination and Application levels are located in the application plane, giving the user a final view of the **IoT** [12]. In service-oriented architecture, ecosystem functions are abstract and accessible via interfaces. The services ensure the stability of the **APIs**, allowing easy integration of new functionalities without affecting the existing system [13]. In the fog-based architecture, four

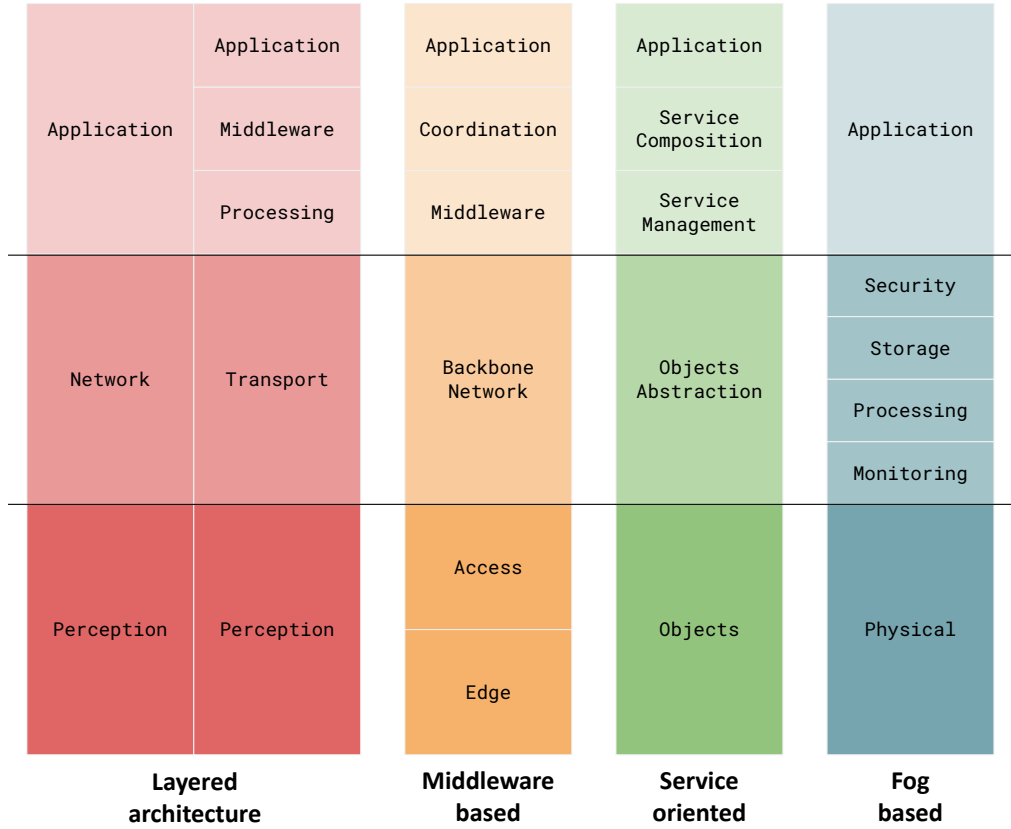


Figure 1.1: Proposed IoT architectures.

levels lie between the physical and transport layers: monitoring, preprocessing, archiving, and security. Monitoring observes sensor data, pre-processing performs operations, storage collects processed data, and security ensures data integrity and privacy [14]. All these **IoT** ecosystem architectures are represented in figure 1.1.

The perception level in the **IoT** is based on sensors and actuators, categorized as low-end, medium- or high-end devices. Sensors collect data, while actuators perform actions. Various sensors detect location, motion, video, audio, light, proximity, environmental parameters and chemicals. These devices have limited operating system, memory, and battery resources. In the data pre-processing layer, there are again technologies with limited storage space, a small processing unit, and basic security to filter and summarize data before sending it to middleware. Pre-processors have more advanced operating systems, more memory, and better batteries than low-end devices. The capabilities of their operating system categorize them as low-end or high-end. Communication technologies for the transport layers include NFC, low-power technology, WSN, and Internet protocols. Devices can use various wireless communication protocols, such as RFID, NFC, Bluetooth, Zigbee, and Wi-Fi. Middleware serves as an abstraction layer that bridges between lower physical levels and higher application levels. Depending on its function, it can be classified as service-based, actor-based, or cloud-based.

1.3 DIGITAL IDENTITY AND THE IDENTITY OF THINGS

Identity is a subjective concept influenced by the environment. It typically refers to who or what something is, often represented by a set of identifiers [15]. Identity can be permanent or temporary depending on the context of the system and has the power to distinguish between entities within a specific scope. It is important to note that there are more identifiers than identities, which in turn exceed entities. While each identity corresponds to an entity, an entity may have multiple identities. Identity is dynamic and can change based on context and purpose. Even within the same scope, an entity can possess multiple identities, highlighting its non uniqueness [16]. Due to the richness of the context, researchers proposed different characterizations.

Identity as a set of partial identities. Clauß and Köhntopp [17] define identity as a combination of personal data associated with a user. An individual's identity is represented as a collection of *partial identities* used in specific contexts. These partial identities may vary depending on the situation (*e.g.*, bank and university). Although users may have multiple partial identities, the repeated use of correlated partial identities may permit to establish reputation. Such et al. [18] formalize these partial identities as follows: given a finite set of attributes $A = \{a_1, \dots, a_n\}$, each with a finite domain $V_{a_i} = \{v_1, \dots, v_k\}$, a set of entities E and entity $e \in E$, a partial identity of e is a vector $I_e = (i_1, \dots, i_n)$ where $i_j \in V_{a_j}$ and $\forall d[d \in E \setminus \{e\} \rightarrow \forall I_d(I_d \neq I_e)]$. Here, V_a represents the values of the attributes $a(a \in A)$, and the set of entities $E(e \in E)$ depends on the context. In addition, the authors argue that the identity of an entity is composed of multiple partial identities. They formalize the identity of an entity e as $I_e = \bigcup I_e^j$, the union of all partial identities.

Identity as a set of attributes. Some authors define the identity of an entity solely by its attributes, without relying on partial identities. For example, Dabrowski and Pacyna [19] conceive of identity as a collection of characteristics (*e.g.*, name, credit card number, passport number) specific to an entity, which distinguish it from others. This entity can be a person, a device or a network service in real life. They argue that the permanence or temporality of the identity depends on the management of the identity, which includes the verification of the authenticity of the entity and the establishment of trust through the authentication of the identity. Similarly, Gomez-Skarmeta et al. [20] conceive identity as a set of attributes that uniquely identify individuals, machines or services with multiple attributes. They propose that each identity be supervised by a trusted entity within its original domain. Attributes can be formally expressed as follows: if $attr_i^e$ denote the i_{th} value of the entity e , then the set of n attributes can be expressed as $ATTR^e = \{attr_1^e, \dots, attr_n^e\}$. Alpár et al. [21] also define identity through attributes, limited to a specific context. They use attribute-based credential technology for adaptable authentication that respects the privacy of the institution. These attributes can be categorized as *identifying* or *non-identifying*, meaning that some attributes are unique to individuals, while others apply to more individuals.

Identity as a set of pseudonyms. Bichsel et al. [22] define identity as a combination of user credentials and self-selected pseudonyms, subject to user-specific policies for accessing resources. These aliases act as a user's public key, but unlike public keys, users can create numerous aliases that cannot be linked to a single secret key. This allows users to use different aliases with different verifiers, all authenticated by the same secret key. However, this presupposes a unique link between pseudonym and user. Camenisch et al. [16] suggest that, apart from their real name, anything can act as a pseudonym for a user, even if it concerns hardware or software. A pseudonym is essentially an identifier, and only the assigning entity knows the true identity behind it. Pseudonyms may be self-appointed or provided by a trusted third party [23]. It is important to stress that the use of pseudonyms as

identifiers protects anonymity in identity management.

The growth of the **IoT** requires a distinctive approach to identity management, different from the traditional systems. The **IoT** features unique characteristics such as scalability, interoperability, and mobility that deeply affect identity management design. The **IoT** involves billions of entities in the digital and physical domains, requiring a highly scalable identity management system. The conventional centralized approach, in which a single entity manages all the identities becomes impractical. Instead, federated identity solutions such as SAML [24] and Shibboleth [25] facilitate the management of different identities from different systems. However, the issue of building mutual trust relationships between different **IdM** systems must be addressed. **IoT** objects vary significantly in communication, information and processing capabilities, leading to interoperability challenges [26]. Unifying identities between different manufacturers, suppliers and standard groups remains a daunting task due to fragmented standards and divergence in vocabulary. An integrative approach is needed to address the heterogeneity of software, hardware, and communications. **IoT** devices, such as vehicles and wearables, are highly mobile and require continuous authentication, authorisation and access control, regardless of their location. The identity management should offer mobility and peer authentication services [27].

1.4 MODERN SOLUTIONS TO AN OLD PROBLEM: THE SHIP OF THESEUS TALE

Aristotle's Law of Identity, a fundamental concept in logic, states that every entity is identical to itself. This, together with the Laws of Contradiction and Excluded Middle, leads to the conclusion that identity is an equivalence relationship characterized by reflexivity, symmetry and transitivity [28]. Subsequently, Wilhelm Gottfried Leibniz introduced the Leibniz Law, known as the Identity of Indiscernibles, which states that no two objects share exactly the same properties [29]. Two principles emerge from the Leibniz Law: the Indiscernibility of the Identical and the Identity of the Indiscernibles. These principles are used to distinguish different entities in both the physical world and the online world of the Internet, because of their intuitive recognition.

Definition 1 (Indiscernibility of the Identical). *If x is identical to y , then x and y share all the same properties. That is: $\forall x \forall y [x = y \rightarrow \forall P (Px \leftrightarrow Py)]$*

Definition 2 (Identity of the Indiscernibles). *If x and y share all the same properties, then x is identical to y . That is: $\forall x \forall y [\forall P (Px \leftrightarrow Py) \rightarrow x = y]$*

However, there are paradoxes in this formalized definition. The well-known paradox of the Ship of Theseus [28] illustrates that when planks and beams are continually replaced in ship A , it becomes difficult to determine whether the ship remains the same after a certain amount of parts has changed. According to definition 1, ship A is identical to itself as long as it maintains the same axes and beams. However, this is not the case. Furthermore, if the replaced planks and beams are joined together in a novel ship B , it is difficult to say that the former ship A and the present ship B are the same, even if they are composed of the same axes and beams, as indicated in definition 2. In other terms, in the **IoT** where entities are dynamic and interconnected, relying solely on attributes is insufficient for identity verification. Unfortunately, we saw in section 1.3 that identity management solutions

in cyberspace are still based on the Leibniz Law, in which people are identified by attributes and authenticated by credentials such as passwords. This approach leads to fragmented identities and increases the risk of personal information theft. Because of this inadequate identity model, users must provide sensitive personal information to prove their identity, and each system operates as a separate identity archive. This results in excessive collection of personal information by unreliable service providers and makes them susceptible to theft due to inadequate security measures in some unreliable systems.

1.5 WHAT YOU SEE IS *NOT* WHAT YOU GET: A MAN-IN-THE-MIDDLE ATTACK APPLIED TO VIDEO CHANNELS

This section provides a tangible example of the potential threat an un-authenticated channel can pose to a digital user. Nowadays, screens are employed as practical interfaces for displaying digital data. Their main function is to simplify the interaction with digital devices by providing visual feedback of their elaboration. On a daily basis, people use these supports in a variety of applications: smartphones, laptops, industrial appliances, and monitoring systems. Looking more closely, the establishment of trust is at the basis for this usage. For example, a user that plugs his laptop to an external monitor assumes that what he sees is exactly the content produced by his laptop. This unconditioned confidence creates a potential weakness that can be exploited by attackers. An attacker could stealthily introduce modifications to the displayed images, thus taking advantage of this misplaced trust. To accomplish this, the attacker can perform a **MITM** attack over the video channel.

MITM is one of the most used and successful class of cyber-attacks and it raises major concerns for security professionals. In its common setup, **MITM** involves two endpoints (*i.e.*, the victims) that communicate with each other and a third party (*i.e.*, the attacker) that places itself in the midst of the communication. Once in place, the attacker can stealthily intercept, modify, change, or replace the communication traffic. **MITM** hence aims at compromising availability (*e.g.*, dropping messages), confidentiality (*e.g.*, eavesdropping) and integrity (*e.g.*, doing modifications) of a communication between endpoints. **MITM** attacks usually exploit poor security design inside well-known communication protocols. ARP [30], DNS [31], DHCP [32], IP [33] are some examples of protocols that have been found weak against **MITM**. Other **MITM** attacks, instead, require a potential attacker to be physically present where the communication is taking place. This is the typical setup for **MITM** attacks implemented on communication channels like Bluetooth [34], NFC [35] and WiFi [36].

On this matter, the video channel between a producer (*e.g.*, a computer) and a consumer (*e.g.*, a screen) of digital video data has been overlooked by both adversaries and researchers. This section proposes itself as a first exploratory work in this branch of studies, demonstrating that **MITM** attacks applied to video channels are an imminent threat that deserves major attention.

We present **MITV**, the first **MITM** attack performed over the video channel connecting a producer and a consumer of digital video data. An attacker can perform a **MITV** by placing herself in the midst of the communication. As a consequence, she can eavesdrop or make real-time modifications to the video stream shared by the two parties. **MITV** only requires to have access to the video channel and to interface with the communication protocol. Hence, it is widely adoptable as it does not require any software or hardware tampering on the parties involved in the communication. While the *modification* of an image could be perceived as a harmless operation, it

can be used to change the perception of reality and to influence the decisional process of a user. An example could be the facilitation of phishing threats [37], where the user is persuaded into clicking on phishing websites. More serious consequences would come from tampering with an industrial monitoring system [38], where an operator could be led to take critical action against false alarms.

To the best of our knowledge, this is the first work proposing a **MITM** attack applied to a video channel. While previous research implemented **MITM** attacks on specific protocols [30, 31, 32, 33] or communication channels [34, 35, 36], we propose **MITV** as a novel class of attacks performed over the video channel. A similar result was shown at the DEF CON 24 conference [39, 40]. Researchers demonstrated an ingenious hack to statically change the images displayed on the victim’s screen. While conceptually similar, this attack is substantially different from our **MITV** attack. In fact, it relies on a malicious software that the victim needs to install and that interacts with the screen to compromise the displayed images. On the contrary, our **MITV** attack only requires to intercept the signal sent over the communication channel. Furthermore, the attack presented at DEF CON 24 can only perform static modifications on the screen, while a **MITV** attacker can produce dynamic changes.

We designed and implemented HackDMI, an effective **MITV** attack conducted over a **HDMI** channel. We used HackDMI for facilitating a phishing attack, where the victim is tricked into accessing a malicious website to steal his personal information (*e.g.*, access credentials). For our experiments, we relied on a HDMI-to-CSI2 bridge module attached to a Raspberry PI 3 model B. The module intercepts the HDMI signal coming from a computer and permits to manipulate it before it is displayed on the screen. We evaluated HackDMI efficacy with quantitative measures. Our implementation is able to elaborate 14 FPS at 720p, effectively modifying the displayed images in a deceptive manner for the user. A demo video showing the complete HackDMI attack is available online*.

1.5.1 SYSTEM AND THREAT MODEL

System model. The system model encompasses the following four components, as shown in Figure 1.2: the producer, the video channel, the consumer, and the user. The producer generates a stream of digital video data, which is transferred to the consumer through a video channel. The video channel could be either *wired* (*e.g.*, HDMI, VGA) or *wireless* (*e.g.*, Wireless HDMI extender adapter dongle). Once received, the consumer processes the video data and displays it to the user. The user then provides an input to the producer based on the consumer’s output. Generally speaking, we assume that the user is a human being capable of perceiving the video data and to act accordingly. However, we do not exclude that also other entities could play the role of the user (*e.g.*, machine learning algorithms).

Threat model. The threat model involves an attacker placed in the midst of the video channel, as shown in Figure 1.3. Depending on the type of the channel, the attack requires a different setup. In the case of a wired video channel, she needs to physically interpose a component between the producer and the consumer. This physical component may be any piece of hardware able to intercept and process the transmitted digital video data. This scenario assumes that the attacker has physical access to the communication channel. A wireless scenario, instead, requires the attacker to be in the communication range of the other components. Moreover, she needs a physical device able to receive and transmit the wireless video data, as well as a jammer to damage the legitimate signal coming from the producer.

*<https://www.youtube.com/watch?v=lvsoJdpNsZA>

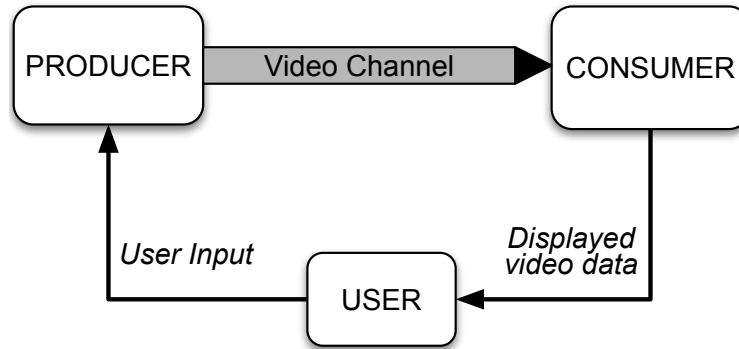


Figure 1.2: Graphical representation of the system model.

In both the wired and wireless scenarios, the attacker can launch several threats, which could be the prelude for other attacks.

DoS – the attacker disrupts the availability of a system by blocking the video transmission. This threat could be particularly effective on environments like power plants or transport stations, where video monitoring systems are extensively used.

Eavesdropping – the attacker aims at accessing the video data, without modifying it. We assume the video data is not encrypted. This assumption, however, does not diminish the extension of this threat as most of the communications between a producer and a consumer are not encrypted (*e.g.*, channel between a computer and its screen).

Modification – the attacker dynamically modifies the video signal. Again, we assume the video data is not encrypted. As an example, the attacker could make appear a green lock on the browser address bar to induce a user to believe the identity of a website is certified.

Once the instrumental configuration and the purpose of the attack are defined, the attacker can establish three different types of communications with the devices involved in a **MITV** attack.

No communication – the attacker does not set up a communication channel with the devices. Once in place, they cannot be further contacted or programmed without being physically accessed.

Local communication – the attacker has only local access to the devices. They can be accessed via a local network through Ethernet connection or via wireless communications (Bluetooth, WiFi).

Remote communication – the attacker can communicate with the devices from remote. She could achieve this by gaining access to an open guest network connection or acquiring an unauthorized Internet access. She could also use a mobile Internet connection.

1.5.2 MAN-IN-THE-VIDEO: A MITM ON VIDEO CHANNELS

MITV is a **MITM** attack where the attacker has complete control over the video channel between a producer and its consumer. Figure 1.4 shows a possible configuration of a **MITV** attack. The user is a human operator that needs to execute a group of tasks. The type of task can vary from monitoring an industrial implant to checking his

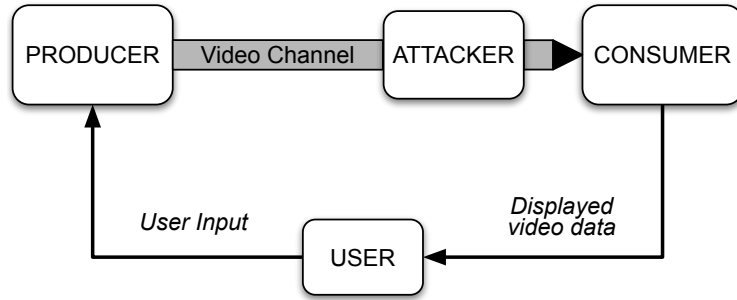


Figure 1.3: Graphical representation of the attacker placed in the middle of the video channel.

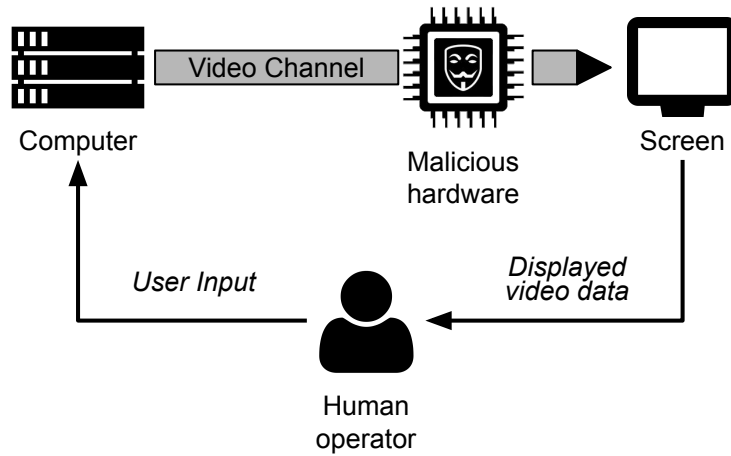


Figure 1.4: Configuration for a Man-in-the-Video attack.

mailbox. These tasks are processed by a computer (*i.e.*, the producer) that generates a video stream as a feedback for the result of the elaboration. The video stream is encoded and transmitted through the video channel using standard protocols (*e.g.*, HDMI, VGA) and it is received by a screen (*i.e.*, the consumer). The channel is the only component with which the attacker must interact. It is worth noting that the attacker does not need to directly interact with the protocol. Once known, she only needs to adapt its hardware with the specific I/O connectors and drivers. The screen works as an interface between the encoded signal sent by the computer and the human operator. The latter uses the information shown on the screen to proceed towards the completion of his tasks. The attacker positions herself in the middle of the video channel to intercept the encoded video stream. To accomplish this, she uses a piece of hardware that is capable of intercepting and modifying a stream of images. Given that a **MITV** attack only exploits the video channel, it is independent from the type of computer and screen involved in the scenario. Subsequently, all computers and screens right now available on the market are virtually susceptible to such attack.

A minimal schematic for a malicious **MITV SoC** is shown in Figure 1.5. The component needs two interfaces to interact with the video stream. The type of interface must be adapted depending on the type of protocol used for streaming the video (*e.g.*, HDMI ports for an HDMI encoded signal). The main purpose of the interfaces is to deal with the decoding and encoding of the signal, so that the processor is completely abstracted from the com-

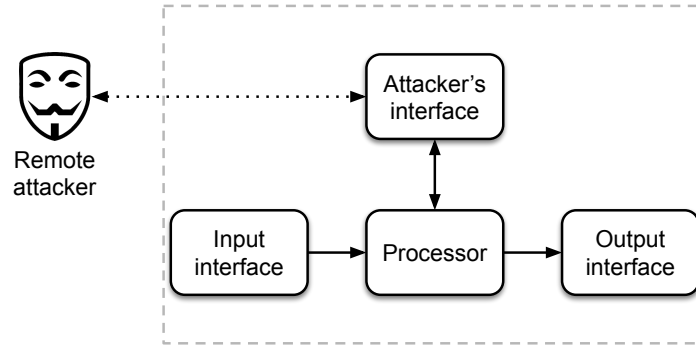


Figure 1.5: Minimal implementation for the hardware involved in a Man-in-the-Video attack.

plexities of the different protocols. The input interface intercepts the signal coming from the computer, decodes it, and passes it to the processor. The processor is the core part of the hardware as it is the one that modifies the video. It must be carefully chosen according to the type of performances that are required. In particular, more elaborated algorithms (*i.e.*, pattern matching or machine learning) require greater computational capabilities to cope with the real-time elaboration of a **MITV** attack. Depending on the type of communication the adversary wants to establish with the processor, an attacker's interface may be necessary. This could be a **SSH** channel or a **REST** service established over the Internet. *No communication* attackers do not need an interface, as they program the processor ahead of installing it. *Local* and *Remote communication* attackers, instead, need an interface for interacting with the processor after the hardware has been installed. Eventually, the output interface takes the result provided by the processor, encodes it with the proper protocol, and sends it through the channel towards the screen.

All these components need a power supply to work. The processor is by far the most energy-demanding part of the hardware. A USB cable would be sufficient for providing energy to a low-power processor. More powerful processors, however, could require a wall power supply.

1.5.3 HACKDMI: A MITV ON HDMI CHANNELS

Our supporting plot. Bob is the CEO of a large corporation. He is the person who makes all the key decisions regarding the company, which includes all sectors and fields of the business. His office is the physical place where most of these activities are conducted. As a support for his work, Bob has a workstation and a screen installed inside his office. The arrangement of those devices inside the room may vary, but it is reasonable to state that the screen is placed over a table while the workstation is concealed under it. The workstation is connected to the screen through an HDMI cable. Bob uses the workstation every day, hence sensitive material concerning the corporation is visualized on the screen.

Mallory is a strongly motivated attacker that is interested in stealing confidential information about Bob and his company. Mallory could be a mercenary hacker hired by competitors for hijacking corporation's secrets or an individual solely interested in acquiring Bob private data. In this setting, we assume Mallory has a double purpose: (i) to eavesdrop on Bob's confidential emails and (ii) to gain access to the company's bank credentials. To achieve her objectives, Mallory decides to implement a **MITV** attack on Bob's workstation. The attack permits to both

eavesdrop and modify the video signal produced by Bob’s workstation. The modification can be used to facilitate a phishing attack aimed at stealing the bank access credentials.

Attack preparation. Mallory starts by acquiring preliminary information about her target. She learns that Bank of America is the reference bank for the company. She also gains information about Bob’s office disposition and connectivity. There are several ways in which Mallory could get this type of information. She could access the facilities by getting a job inside the company or by working as a cleaner. She could also pay an employer for receiving sensible data from an insider. The office disposition permits Mallory to evaluate the feasibility of the attack. Given that the workstation is hidden under Bob’s desk, she decides to place the **MITV** hardware on its HDMI output. This way, it will not be visible as it would be if placed behind the screen. Information about the connectivity lets Mallory decide which type of communication she can establish with the hardware. Again, depending on the scenario, different solutions are available. In case of *No communication* scenario, she could eavesdrop the video data by storing the screenshots locally and program the hardware to perform predefined modifications on the video. In case of *Local* or *Remote* scenario, instead, she could eavesdrop the video data by sending the screenshots through the network and she could adapt the modification on-the-go. For this attack, Mallory decides that the *No communication* strategy is sufficient.

After this preliminary study, Mallory prepares the phishing website for stealing the credentials. She creates a copy of Bank of America website using a tool like Htrack [41]. She then acquires a domain in a web hosting platform to put the copy online. The web hosting can be either free or paid, but it should provide HTTPS websites to simplify the software implementation of the attack. Mallory chooses a free hosting from ooowebhost [42] and creates the website `https://lolololololololololololol.000webhostapp.com/`^{† ‡}. The unusual name serves as base for the software part of the attack (Section 1.5.3).

To make Bob visit the phishing website, Mallory prepares a phishing email that will be sent to his inbox. The email is created using a standard template that can be found on the Internet. The email will ask Bob to accept the new terms and policies by visiting the embedded URL and logging into the company’s bank account. The URL points to Mallory’s phishing website. The software implementation will take care of hiding the phishing URL behind the legitimate one.

Regarding the hardware, Mallory opts for a Raspberry PI model 3. She chooses a HDMI-to-CSI2 module to integrate the Raspberry PI with an HDMI input port. These two components, as well as their setup will be described in greater detail in Section 1.5.3. She programs the Raspberry PI to store screenshots of the video stream and to perform the desired modifications. More details about the software implementation will be provided in Section 1.5.3. Eventually, Mallory needs to physically access Bob’s office to install the Raspberry Pi. This is achievable by using the same strategies listed in the preliminary part. The installation process is straightforward and requires to: (i) detach the HDMI cable behind the workstation and connect it to the Raspberry Pi; (ii) connect the HDMI port of the workstation to the HDMI-to-CSI2 module through a second HDMI cable. Given its simplicity, the task can be completed in a small amount of time, indicatively a couple of minutes. While the physical access to

[†]The domain has been flagged as a phishing fraud and paused shortly after recording the demo footage. This does not represent a limitation for the attack as Mallory could have used a domain hosted on a server she owned.

[‡]We could have created a local phishing website to avoid ethical implications. We decided to use a public hosting to test our attack on a real implementation. The website only hosted a copy of the homepage of Bank of America, hence it was not fully functional. We used it only to demonstrate the real-time modification operated by our appliance.

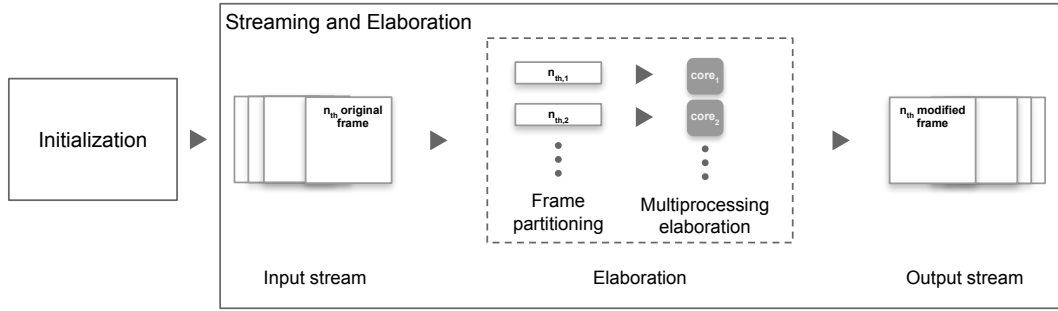


Figure 1.6: Software pipeline for HackDMI.

the office is a strong assumption, the ease of installation makes this attack realistic and applicable.

Hardware configuration. Bob has a HP Z230 Tower Workstation connected through an HDMI cable to an HP EliteDisplay E230t. Mallory chooses the Raspberry PI model 3 as her supporting hardware. It mounts a Broadcom BCM2837 SoC with a quad-core ARM Cortex-A53, 1.2GHz cluster. She then opts for the Auvideo B101 HDMI-to-CSI2 bridge module [43] as HDMI input interface for the Raspberry PI. It supports 1080p25 HDMI input through a Toshiba TC358743XBG chip. The connection is done using a 15-pin FPC flex cable (pins on top) that is inserted on the CSI-2 camera port of the Raspberry PI.

The Raspberry PI natively supports the Toshiba chip through the Device Tree overlay *tc358743*. However, there are some steps required for making it operational: (i) Enabling the camera module inside *raspi-config*. (ii) Activating the tree overlays by adding *dtoverlay=tc358743* at the end of */boot/config.txt*. (iii) Reserving memory for the camera buffer by adding *cma=96M* at the start of */boot/cmdline.txt*. (iv) Rebooting. The module can be configured from the Raspberry PI command line using the *v4l2-ctl* tool. It permits to control Video4Linux devices that implement the V4L2 API [44]. In particular, the equipment must be configured to work as a proxy between the workstation and the screen. In the first place, the HDMI output resolution of the Raspberry PI must match the one coming from Bob's workstation. It is possible to force the Raspberry to adopt the same resolution by adding an *hdmi_mode* to */boot/config.txt* and rebooting. An extensive documentation on the matter is available at the following link[§]. Afterwards, the B101 module requires an **EDID** file to function properly. The **EDID** describes the characteristics of a screen and permits to set the right resolution and frequency of the video signal. To mimic Bob's screen, Mallory can extract the **EDID** file from a similar monitor. It is sufficient to attach the monitor to a Linux computer and to execute the following command.

```
xrandr --props
```

The command prints the details of all the screens attached to the computer, including their **EDID** data. The printed data can now be stored inside *edid.txt* and can be loaded on the module using *v4l2-ctl* on the Raspberry PI.

```
v4l2-ctl --set-edid=file=edid.txt
```

The module can now be set to receive the stream from Bob's workstation.

[§]<https://www.raspberrypi.org/documentation/configuration/config-txt/video.md>

```
v4l2-ctl --set-dv-bt-timings query
```

This procedure makes the stream available to the Raspberry PI as if it were captured by a Pi Camera [45]. To modify it, it is now sufficient to use a programming language that natively interfaces with the camera. The software implementation is described in greater detail in Section 1.5.3.

Software implementation. The software implementation should find a well-balanced trade-off between the quality of the modification and the speed of elaboration. The objective of the attack is to hide a phishing URL under a legitimate one. To achieve this, a **MITV** attacker substitutes the phishing URL every time it appears in the video stream. The key part of the software is the algorithm that detects which areas of the frame must be modified. In general, this is a pattern matching problem where the image of the phishing URL is the pattern that needs to be found inside the frame. Mallory chooses Python 3.8 [46] as her supporting language. Python is a powerful and versatile scripting language that has a vast array of libraries at its disposal. In particular, OpenCV [47] completely abstracts the capturing process from the camera. and proposes a set of advanced pattern matching functions. However, given the high computational power required (*e.g.*, ORB [48], Template Matching [49]), Mallory designs a custom pattern matching algorithm to better suit her hardware possibilities.

The starting point is the consideration that the algorithm for retrieving the contours bounding boxes in an image is an efficient task. It is then possible to efficiently apply this function to detect a particular sequence of bounding boxes. For example, creating a website domain with a particular choice of letters (*e.g.*, *lolololololololololol*) generates a unique sequence of bounding boxes. That particular fingerprint becomes a pattern that the attacker uses to locate the phishing URLs with high confidence. Figure 1.6 shows a diagram of the software pipeline. The overall process can be decomposed into two main parts:

Initialization, where the HDMI-to-CSI2 module is loaded, the I/O buffers are created and a different process for each core is started;

Streaming and Elaboration, an endless loop that manages the I/O operations and feeds the active processes with the data to elaborate. Each process then works on its piece of frame in parallel.

Initialization. This phase (Algorithm 1.1) is preparatory for the code execution. During this part, the B101 module is loaded (line 1) and it is used to grab *height*, *width* and *depth* of the frame (line 2). The dimensions are used for creating the I/O stream buffers (lines 3-4). The output buffer must coincide with the frame-buffer used by Linux systems to write on screens. On a standard Raspberry Pi OS, this is a file that can be found under the path */dev/fbo*. Opening it as a common file permits to write inside of it. For better performance, it is also possible to map the frame-buffer directly into the working memory. The initialization concludes with the creation of separated elaboration processes (line 7). Each process receives a pointer to the buffers (lines 8) and the boundaries of its competence region inside them (line 9). In this way, the buffers are evenly partitioned among all the processes available, thus effectively distributing the elaboration burden. The number of parallel processes that can be used is directly related with the number of CPU cores available. In Mallory's case, for example, the maximum number of processes is 4.

Streaming and Elaboration. After the initialization part is performed, the software is ready to stream and modify the video signal (Algorithm 1.2). The frames are periodically acquired using an infinite *while* loop (line 1). The current frame is stored inside the *input* buffer using the *capture* object (line 2). At that point, the processes start the elaboration over their piece of frame (line 4). The processes already obtained a reference to the *input*

Algorithm 1.1 Pseudocode for the Initialization phase.

```
capture ← getCaptureModule()
 $h, w, d \leftarrow \text{capture.getFrameDimension}()$ 
input ← [ $(h \cdot w \cdot d) * \text{byte}$ ]
output ← [ $(h \cdot w \cdot d) * \text{byte}$ ]
processes ← []
for  $n < \text{getNumberOfprocessors}()$ 
  process ← createNewProcess()
  process.linkToBuffers(input, output)
  process.assignPartition( $h, w, n$ )
  processes.append(process);
end for
```

Algorithm 1.2 Pseudocode for the Streaming phase.

```
while True
  input ← capture.getFrame()
  for process ∈ processes
    process.elaborate()
  end for
  for process ∈ processes
    process.waitElaboration();
  end for
end while
```

buffer during the initialization part. The main process then blocks and waits for all the processes to finish their elaboration (line 7). When this happens, the cycle restarts and requests another frame from the input stream. The result of the elaboration is directly written in the *output* buffer by each worker separately to improve performances.

Each process elaborates its specific part of the frame (Algorithm 1.3). The algorithm is composed of a sequence of 8 distinct operations. After finding the bounding boxes for all the contours inside the image (line 2), the bounding boxes are filtered according to their dimension (line 3). The URLs that need to be modified are written using a small font. Hence, exceedingly large boxes do not need to be processed and can be unlisted. The algorithm proceeds by ordering the boxes in rows by using their y -position on the screen (line 4). Two boxes are considered in the same row if their vertical disposition overlaps. Each row is then separated into words (line 4) by grouping closer boxes using the x -position. The proximity for the grouping is a parameter that is tuned after performing different trials.

The specific pattern is then searched inside the resulting list. Depending on which name the attacker chose for the phishing website, the pattern changes. In Mallory's case, the word is *lololololololololololol*. It generates an unambiguous sequence of 23 consecutive bounding boxes that are alternately *tall-and-narrow* and *short-and-wide*. The length of the sequence is carefully calculated to match that of the original URL. Furthermore, the spacing between the letters renders the detection of the bounding boxes effective even for smaller fonts. As a consequence,

Algorithm 1.3 Pseudocode for the Elaboration phase.

```
while True
    boxes  $\leftarrow$  input.findBoundingBoxes()
    boxes  $\leftarrow$  filterByDimensions(boxes)
    rows  $\leftarrow$  separateIntoRows(boxes)
    words  $\leftarrow$  []
    for row  $\in$  rows
        words.append(separateWords(rows))
    end for
    words  $\leftarrow$  filterByLength(words)
    patterns  $\leftarrow$  []
    for word  $\in$  words
        if patternDetected(word)
            patterns.append(word);
        end if
    end for
    for pattern  $\in$  patterns
        modifyImage(pattern, output);
    end for
end while
```

the algorithm can confidently filter from the list all the words that are not 23-characters long (line 9). To account for bounding errors, it is also possible to include words with a length close to the expected one.

Eventually, the filtered list is scanned for those words that present the pattern (line 12). In Mallory’s example, the algorithm searches for a binary pattern where *tall-and-narrow* boxes and *short-and-wide* boxes alternate with each other. Again, to account for bounding errors, it can tolerate small deviation from the original pattern. This last operation isolates all the occurrences of the phishing URL inside the frame. At this point, it is sufficient to substitute the text inside each of these boxes to hide her URL (line 16).

1.5.4 ANALYSIS AND RESULTS

To evaluate both the efficiency and the feasibility of HackDMI, we collected both quantitative and qualitative measures.

To assess HackDMI efficiency, a good approach is to consider the **FPS** that the attack is able to guarantee. This quantity is crucial, as a minimum number of FPS is required for a video to be perceived as smooth. In particular, 24FPS has been the standard frame rate for movies and videos for long time [50]. While the number was merely chosen for its divisibility, it was also considered fast enough to avoid flicker. Using this number as a reference for a good video quality, it is possible to comment the results shown in Figure 1.7. In particular, the graphs show the **FPS** generated during the execution of the algorithm with various resolutions. To better understand the impact for each of the steps involved in the frame elaboration, we gradually added them while continuously

recording the output **FPS**. Figure 1.7a shows the performances with no parallel processes. Figure 1.7b, instead, plots the performances with the aid of 4 parallel processes. It is immediately clear the impact of parallel execution that demonstrates its usefulness in this use case. Despite this, however, only the 640x480 60FPS (480p) resolution achieves optimal frame-rate (according to the chosen reference), while the 1280x720 60FPS (720p) and the 1920x1080 30FPS (1080p) are still below the threshold. It is interesting to note, though, that part of the computational cost is due to the first step of the algorithm (*i.e.*, SS - SimpleStream). During this step, the algorithm is only supposed to acquire the frame from the module and to copy it into the output frame. In the 720p experiment, this step outputs around 30FPS, while in the 1080p around 15FPS. These values are much lower than the original frame-rate of the stream, indicating that the acquisition and copying is incurring in an overhead. This leaves margin for further improvement, as a more efficient interface with the stream could reduce the cost.

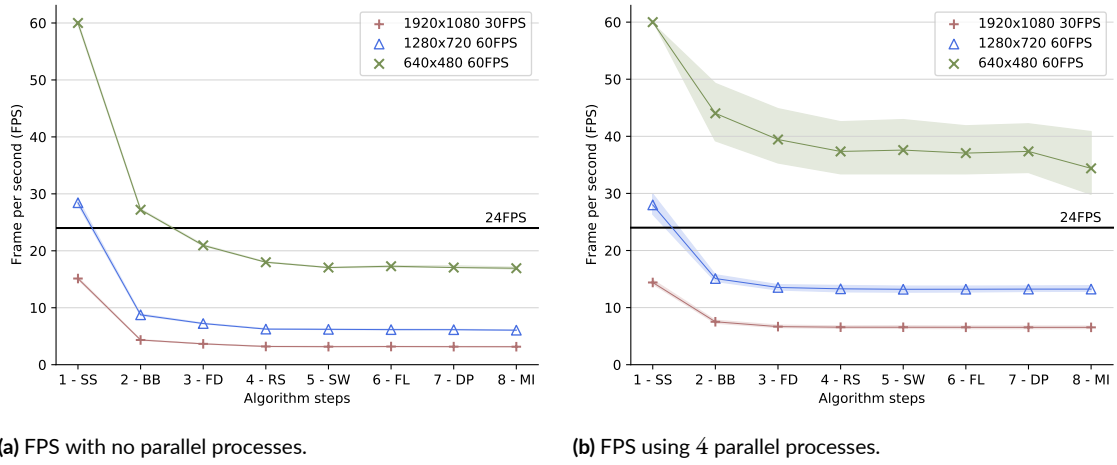


Figure 1.7: FPS generated at different stages of the elaboration process (SS - SimpleStream, BB - BoundingBoxes, FD - FilterDimensions, RS - RowSeparation, SW - SeparateWords, FL - FilterLength, DP - DetectPattern, MI - ModifyImage). The graphs plot the averages of 5 different experiments. The shaded area represents the standard deviation.

To qualitatively evaluate the feasibility of HackDMI, two videos were recorded that show the phishing attack with and without the HackDMI instrumentation connected to the victim's screen. Both of them are available online[¶]. Finally, a complete demo video of the phishing attack was recorded and made available online^{||}. The resolution for producing these videos was set to 720p. The choice of this value was driven by the quantitative results. The recordings are useful to evaluate how much discrepancy exists between the unmodified stream and the modified one. The perception of the lower frame-rate is different from person to person and it depends on different circumstances (*e.g.*, age of the victim, experience with digital devices).

[¶]https://mega.nz/file/uLMUxDRK#KsDTc9QP-a-e5KYitEzfDthSNzJc_sB-sIp1X3SuJ_g
https://mega.nz/file/OpSWSTAS#2lzw_YcyMnmSJ8hjQbXuvcpva0ut2Jz7TEUW-9sJQQw
^{||}<https://www.youtube.com/watch?v=lvsoJdpNsZA>

1.5.5 RELATED WORKS

MITM is a well documented class of attacks. The concept of stealthily intercepting communications has been applied to different protocols and technologies over the years. An extensive survey on **MITM** attacks is proposed in [51]. This work identifies three ways of characterising MITM attacks: *impersonation technique*, *communication channel* and *location of attacker*. A MITM attacker can find herself in three different positions *w.r.t.* the target [52]: (i) *local*, when attacker and target are on same network; (ii) *local to remote*, when the attacker reaches the target through a gateway; (iii) *remote*, when the attacker remotely connects to a local controlled device. The *impersonation techniques* are the modes in which attackers convince their victims to trust the communication. Spoofing-based **MITM** are attacks in which the attacker intercepts a legitimate communication by means of a spoofing attack [30, 31, 32, 33]. SSL/TLS MITM is a form of active interception, where the attacker establishes two separate SSL connections with the victims and relays messages between them [53, 54]. BGP MITM consists in an IP hijacking attack where the attacker makes stolen traffic to be eventually delivered to the original destination [55]. FBS-based MITM uses fake Base Transceiver Stations (BTS) to make victims connect to them [56, 57]. MITM attacks can also be grouped by the *communication channel* on which the attack is performed. Some examples are Bluetooth [34, 58, 59, 60], NFC [35], WiFi [36], mobile communications [61, 62], ad-hoc networks [63, 60].

Our work can be classified as a MITM attack. However, its field of application completely differs from what has been seen and proposed to date. To the best of our knowledge, this is the first time a MITM-based threat is applied to a video channel (e.g., HDMI, VGA).

An extensive literature is available on electromagnetic (EM) side channel attacks. These works reconstruct the images projected on a screen by intercepting the EM waves emitted by the screen itself. Lemarchand et al. [64] propose an upgraded EM side channel attack able to automatically reconstruct the intercepted data. Zhang et al. [65] combined the radiated fields of a LCD screen with the transfer function of the test probe to extrapolate the relationship between the RGB signal and the intercepted one. Elibol et al. [66] use low-cost and all-in-one mobile receiver to capture screen emanations from a long distance (up to 50 meters).

The work presented in this section is substantially different from these works. Previous references uses EM sniffers and antennas to intercept and reconstruct the signal propagated by a screen. Our system, instead, places a piece of hardware in the middle of the transmission to intercept it. The different position permits our appliance to apply real-time modification to the transmission, while other works are limited to eavesdropping it.

The security of video transmission to monitors is a rather uninvestigated avenue. Some research has been conducted on the Consumer Electronics Control (CEC) protocol embedded into the HDMI standard [67]. CEC lets users control multiple HDMI interfaces using a single remote device. Different vendors created custom implementations of CEC and some of them have been found vulnerable [68, 69]. In [69] researchers show how current insecure CEC practices open the possibility for an attacker to perform malicious analysis, eavesdropping, DoS and even facilitate other well-known existing attacks through HDMI. Instead, the work in [68] exposes several vulnerabilities by applying fuzzing techniques to the CEC implementations of major HDMI vendors. To secure the video stream against the piracy of copyrighted contents from online streaming platforms and DVD Blu-rays, the HDMI standard incorporates the High-bandwidth Digital Content Protection (HDCP) [70]. While this feature could theoretically be applied to the video signal between computers and screens, this is never the case in normal life scenarios.

Our work is completely detached from these as it does not target any protocol-specific security hole, but operates on

the data sent to a monitor. Furthermore, the HDCP functionality is only offered by HDMI connection to protect the streaming of copyrighted material. Thus, it does not constitute a mitigation for the attack here presented.

A conceptually similar attack to monitors has been shown during the DEF CON 24 conference [39, 40]. Researchers showed that by targeting the monitor's embedded computer they were able to remotely tamper with the pixels' values using a blinking pixel visualized on that same monitor.

While the final result is (seemingly) the same, the work here presented is substantially different: (i) the DEF CON attack requires the victim to install a malware inside her computer, while a MITV attack only requires access to the communication channel; (ii) the DEF CON attack depends on the targeted class of hardware, while a MITV attack is virtually applicable to all type of communications established between a computer and its screen; (iii) the modification performed in the DEF CON attack on the screen is static, while a MITV attack adapts with the video and applies dynamic modifications.

1.5.6 DISCUSSION

Hardware limitations. The potential of a MITV attack heavily depends on the type of hardware employed. The modification used in this work has demonstrated to be effective even with a limited hardware such as the Raspberry Pi. However, a more performing hardware would permit the use of more stable and powerful solutions. An example is *Template matching*, an OpenCV functionality that enables to find patterns inside an image. Another solution would have been *Machine Learning* for detecting specific patterns. These algorithms would enable a much more robust pattern detection *w.r.t.* the one proposed in this section. To augment the hardware capabilities, the attacker could consider buying the latest Raspberry PI model 4 that offers enhanced performances *w.r.t.* its older version. Another possibility is the use of multiple Raspberry PIs to speed up the elaboration. The cluster would consist on one master and several slave Raspberry Pis. The master would manage the video stream and distribute the burden of calculation between the slaves. Another solution could be the use of the Raspberry PI as a bridge to transmit the stream to a more powerful machine for modification. For example, the stream could be transmitted to a cloud platform and modified using complex machine learning algorithms. While this solution would permit virtually infinite computational resources, it needs to carefully consider network latency for providing a real-time stream.

Case study considerations. We chose phishing as an example to demonstrate the feasibility and the potential of MITV attacks. We used it to steal the credentials for accessing a bank account. The same result could be accomplished using a keylogger. A keylogger is a hardware or software tool that is capable of logging the computer keyboard. By means of that, the attacker can secretly intercept the typed characters. However, while the result is the same, the requirements and the dynamics of the attacks are substantially different. The installation of a keylogger requires physical access to the computer itself. A software keylogger needs the attacker to possess access credentials to the system and potentially to deal with installation policies. A hardware keylogger needs the attacker to disassemble the computer shell. Instead, the attack presented in this section only demands access to the external HDMI cable, thus requiring less complexity *w.r.t.* the keylogging alternatives. Another aspect that is worth to be discussed is **2FA**. Modern home banking systems utilize **2FA** for enhanced security. **2FA** requires to present at least two pieces of evidence for authenticating an user, thus dramatically lowering the risk related to credentials' sniffing. Most common **2FA** services employ either a token or an app confirmation to complete the authentication

phase. The attack presented in this work did not consider **2FA** for the sake of simplicity. It is possible, though, to design more complex setups to overcome this limitation. Generally speaking, the updated version of the attack would use the phishing website as a counterpart to monitor and register the actions of the victim. Each action would be replicated in background on the original website. **2FA** would be eluded by waiting for the victim to perform the second action (*e.g.*, token input or app confirmation). Note that the attacker has complete control over the final result of the operation. For example, whenever the victim performs a bank transfer she can easily change the amount and the destination of it.

As a side note, we want to underline that these considerations are specific to the scope of the application here presented (*i.e.*, phishing of a bank account). We emphasize that further observations or objections related to the practicality of the phishing application must not be seen as general limitations to the overall **MITV** concept.

Future works. We are planning to improve the frame rate reached by our prototype by either refining the existent workflow or changing the supporting hardware with a more suitable board. Another interesting path of research would be the integration of our attack directly into the graphic card of a computer. HDMI cables transmit audio data together with the video stream. As a future development, we could integrate the modification of the audio stream along with the one on the video stream.

Measures against the attack. **MITV** attacks are subtle in the sense that they completely bypass the common security measures present inside a computer. The simplest and most effective countermeasure is the awareness of the user. **MITV** attacks require an external device to be connected to the HDMI connection. A physical check prior usage performed by the user would drastically reduce the probability of successful attack. In case the direct check is not an option (*e.g.*, the cable is not accessible), the user would need to encrypt the transmission between the computer and the screen. Current appliances do not support encryption of the video data flowing from a computer to its external screen. Hence, the only way to achieve this is by means of external devices that take care of the cryptographic transformations.

Notice that the attack is possible because of the lack of authentication of the communication channel. A proper identification of the channel and/or the monitor would prevent such inconvenience. In Chapter 2 we will introduce cryptographic secrets as a simple and effective solution to authenticate entities. In particular, Section 2.2 proposes a fuzzy extractor to derive stable cryptographic secrets from circuit noise. Such technology would enable, for example, to implement a cheap and effective cryptographic storage by using hardware already embedded inside the monitor (*e.g.*, RAM).

2

Traditional single-attribute identification

This chapter introduces the traditional single-attribute identification techniques. Digital identities rely on cryptographic methods to secure their usage. There are two main families of cryptographic solutions, namely symmetric and asymmetric encryption [71]. In symmetric encryption, there is a single secret key that is used in both the processes of encryption and decryption. On the contrary, asymmetric encryption involves the usage of two distinct keys for encryption and decryption. Passwords are an example of symmetric encryption, where two parties first agree on a shared secret, and then use it to authenticate with one another (*e.g.*, through encryption/decryption of predefined message). Digital signatures are instead an example of asymmetric encryption. A public repository stores a list of decrypting keys and certifies their association to specific identities. An entity can then authenticate by using the corresponding secret encrypting key.

Section 2.1 showcase a classical implementation of asymmetric cryptography for authenticating a group of **IoT** devices. In this case, the authentication forms a strong base for an innovative, autonomous attestation system for restricted devices. Instead, section 2.2 delves into the problem of storing sensitive material inside unprotected memories. **IoT** devices are low-end appliances that cannot integrate expensive, isolated memory units. Recently, researchers proposed **PUFs** as potential solutions to this problem. Section 2.2 proposes a secure fuzzy extractor that permits to safely and consistently extract a secret key out of a fuzzy source.

2.1 HOLLA: HOLISTIC AND AUTONOMOUS ATTESTATION FOR IoT NETWORKS

IoT refers to a category of small independent devices that, when connected to a network, can autonomously collaborate to accomplish complex tasks [72]. The widespread use of **IoT** technologies attracted the attention of adversaries, leading to the development of a broad class of attacks. These attacks are often partitioned into two groups: (i) *software attacks* that install malicious software inside the device [73, 74, 75]; (ii) *physical attacks* that tamper directly with the hardware [76, 77, 78, 79]. While *software attacks* are remotely executed by leveraging classic hacking techniques, *physical* ones require an adversary to remove a device from a network for a non negligible amount of time (*e.g.*, 10 min [80, 81]). In this scenario, **RA** is a major solution for validating the integrity (*software* or *physical*) of remote devices [82]. The classic **RA** scheme [83] involves a trusted entity (*i.e.*, *Verifier*) that challenges a remote device (*i.e.*, *Prover*) to provide a measurement of its current status. Over the past few years, researchers proposed **CRA** schemes that better fit the *mesh-like* environment of **IoT** (*i.e.*, networks with devices communicating only with physically-close neighbours). However, the **IoT** world is increasingly moving from *mesh-like* to *Internet-like* networks [84, 85]. Here, the concept of physical neighbour vanishes and current **CRA** schemes show limitations in terms of scalability and security. We consider the adoption of *Internet-like* networks prominent in light of the research on new energy-save Wireless protocols (*e.g.*, 6LoWPAN [86], Thread [87]) that promise to connect many more **IoT** devices to the Internet itself.

In light of these considerations we propose HolA, the first Holistic and Autonomous Attestation protocol for *Internet-like IoT* networks that: (i) guarantees an effective, efficient, and scalable periodic attestation of the whole IoT network; (ii) makes the IoT network resilient to the well-known attacks targeting *mesh-like* networks and the new ones addressing the *Internet-like* networks. We implemented HolA on real devices equipped with a trusted anchor [88] for storing keys and performing cryptographic operations (*i.e.*, 5 Raspberry Pi 3 and a Raspberry Pi

o for performance reference). We also evaluated HolA performance in a large scale simulated network (*i.e.*, 1M devices) through Omnet++ [89]. To validate our approach, we conducted several attacks in both real and virtual scenarios, encompassing software tampering, lost packets, and corrupted devices.

2.1.1 BACKGROUND

Remote Attestation. RA schemes consist in protocols that permits the verification of a remote entity. Usually, RA schemes involve two distinct roles: *Verifier* and *Prover*. The *Verifier* is considered trusted and is usually physically protected from attacks (*e.g.*, a remote server). The *Verifier* duty is to verify the integrity of a *Prover* that may be corrupted (*e.g.*, due to a malware). RA schemes require a *Verifier* to start the protocol by sending a challenge to the *Prover*, which measures some properties of its state (*e.g.*, compute a hash of a piece of software) and returns a report. Then, the *Verifier* can validate the *Prover* status by matching the returned report with a database of correct measurements. In **IoT** scenarios, it is a common practice to perform *single-device RA* and *collective RA*. In the former, any network device can play the role of the *Verifier* and issue a challenge to another network device, *i.e.*, a *Prover*, to attest its status. In the latter, only a device from a set of predefined ones can verify the current status of all the other network nodes, *i.e.*, *Provers*, and generate a cumulative report. The classic approaches involve static measurements, such as software fingerprint or hardware integrity [90]. More recently, researchers proposed dynamic RA [91, 92], which tries to attest the run-time properties of the *Prover*, such as its execution-paths.

Trusted Anchor. Modern RA schemes require nodes mounting specific hardware, called *trusted anchor*, that correctly implement minimal hardware features for attestation [83]. The nodes use Read-Only Memory (ROM) and Memory Protection-Unity (MPU) to partition the device memory into two zones: (i) *untrusted*, containing general purpose software; (ii) *trusted*, a protected memory region shielding sensitive information, such as cryptographic algorithms, keys, and secure random number generators. In short, the *trusted anchor* guarantees that only the protocol code accesses the cryptographic keys, and the node is booted correctly. Recent works show that Off-The-Shelf IoT devices already provide *trusted anchors* with a minimal hardware features set [93].

Chord.

Chord [94, 95, 96] is a **DHT** protocol for managing distributed hash tables. Chord uses consistent hashing for arranging the nodes in a ring and for distributing the keys among them. In Chord, each node is identified by an *m-bit* number computed by a hash function. The Chord ring stores a piece of data P inside the first node whose identifier is greater than or equal to the P identifier. For instance, a node D contains all the piece of data whose ID is between D and the node before D . Using these identifiers, nodes are linked to their predecessors and successors, thus creating a ring. To improve resilience, nodes maintain a list of successors called *successors list*. The routing of messages around the ring is made efficient by the introduction of the *fingers table*. Finally, any node has a *finger table*, that contains m entries whose i -th element is a reference to the 2^{i-1} position ahead the current node. that allows an average routing complexity of $O(\log_2(n))$ and renders the operation scalable *w.r.t.* the number of nodes in the network. Fig. 2.1 depicts a simple ring in which a node N is linked to its successors (solid-arrows) and to its fingers (dashed-arrows).

Having the nodes and data organized in a ring makes possible to implement a look up function as follows: (i) a node A sends a request with the data ID to its *successor* B , (ii) if B contains the data (*i.e.*, $ID < B$), it is returned to A , otherwise it forwards the request to its *successor*. The step is repeated till finding the ID.

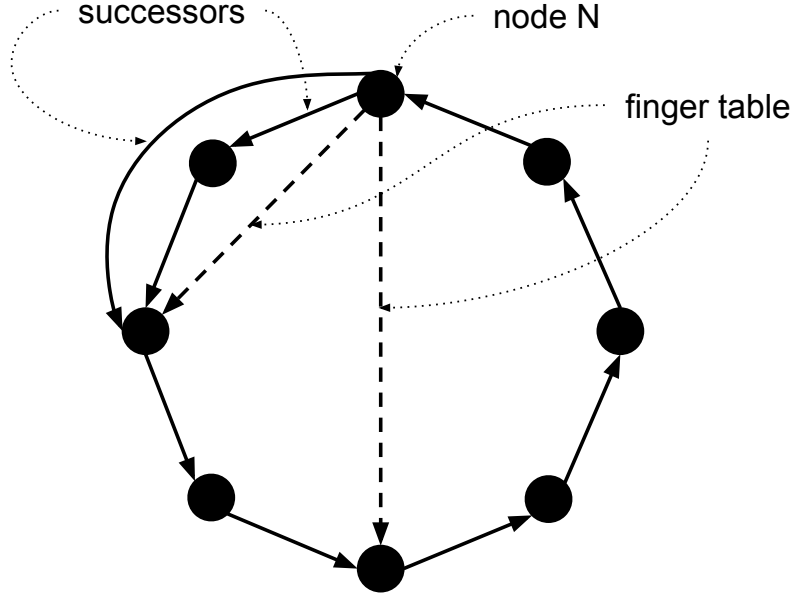


Figure 2.1: Simplified Chord organization. Each node is connected to its direct *successors* and contains a *finger table* to speed up seeking operation.

This naive look-up Chord implementation has a complexity which is linear with the number of nodes placed inside the ring. Researchers improve the performance of Chord with the introduction of a *finger table* that contains additional routing information. The *finger table* has m entries called *fingers*: the i -th finger is a reference to the 2^{i-1} position ahead the current node. As a result, the *finger table* allows an average searching complexity of $O(\log_2 N)$, making the look up operation scalable with respect to the number of nodes N . Fig. 2.1 depicts a simple ring in which a node N is linked to its *successors* (solid-arrows) and to its *fingers* (dashed-arrows).

Chord permits dynamicity in the network by introducing three maintenance tasks [96]: (i) the *join* task, where an outside node contacts a member of the ring to join the topology; (ii) the *stabilize* task, where a node contacts its direct successor to check its presence and possibly adjust disruptions using the *successors list*; (iii) the *rectify* task, where a node receives notification of presence from its predecessor. The *join* task is performed only when a node is entering the network, while the *stabilize* and the *rectify* tasks are periodically executed by all the nodes to maintain the ring topology.

HolA uses Chord as transport layer of the ISO/OSI model to connect the **IoT** nodes in a ring. Since Chord does not provide security by design, HolA introduces a new security layer on top of it. We then perform our attestation protocols over this strengthened version of Chord.

2.1.2 ASSUMPTIONS

System Model. HolA focuses on *Internet-like* networks where devices are equipped with a trusted anchor. Devices communicate with each other over a secure and reliable channel. The security of the communication is guar-

anteed by the adoption of known protocols (*e.g.*, Diffie-Hellman [97]) on top of the Internet ones (*e.g.*, WiFi [98], 6LoWPAN [86] and TCP/IP [99]), while the reliability comes from the TCP properties. Each device is uniquely identified by a certificate signed by a CA controlled by the network owner. The device private key is stored within the trusted anchor, while the public key is shared with other nodes to issue a secure channel. Finally, we assume nodes are already equipped with countermeasures against side channel attacks and having clocks loosely synchronized, as already assumed by previous works [81, 80, 100].

Threat Model. The goal of an attacker is to gain control of a network device and compromise it, meanwhile preventing its detection from the rest of the network. To achieve this goal, the attacker can use two different strategies:

- *Software attack* (*i.e.*, A_{sw}): working from a remote location, the attacker can gain control of the untrusted zone of one or more network nodes through classic exploitation techniques, but not of the trusted one. Moreover, she can gain control of one or more network infrastructure nodes (*i.e.*, Dolev-Yao model [101]).
- *Hardware attack* (*i.e.*, A_{hw}): the attacker can gain control of both the trusted and untrusted zone of one or more network nodes by manually tampering with the hardware node. Thus, the attacker needs to be in a physical range with the device, to remove it from the network for a time T_a (*e.g.*, 10 min [76, 77, 78, 79, 80, 81]) and compromise it.

Adopting the above-mentioned strategies, the attacker can complete two attacks:

- *Injection attack*: through a A_{hw} , the attacker can inject a compromised device into the network.
- *Compromising attack*: through either a A_{hw} or a A_{sw} , the attacker can compromise a node already belonging to the network.

To inject a compromised device into the network, an adversary needs a valid certificate. Thus, she can either obtain a valid certificate from the CA or steal the certificate from another network node. While the first option is unfeasible, the second one is doable provided that the original owner is excluded from the network. In addition, the adversary can rely on A_{hw} to manipulate the network infrastructure and tamper with the protocol. To compromise a network node, the attacker can rely on A_{hw} or on A_{sw} . Through A_{hw} , the attacker physically removes a node from the network for a time T_a , installs malicious code inside it, compromising the trusted zone, and makes the node rejoining the network. Moreover, the compromised node might manipulate the network traffic to prevent its detection. Through A_{sw} , the attacker reaches a network node from a remote location, installs malicious code inside it, compromising the untrusted zone, and manipulates the network traffic, either from the compromised node or from a network infrastructure node, to prevent the node detection. In this work, we do not consider destructive DoS attacks that utterly interrupt any communication.

2.1.3 MOTIVATION

CRA Limitations in Internet-like Networks To motivate our claim, we discuss here the impact of deploying the following three different CRA schemes in an *Internet-like* network: SANA [102], one of the most scalable protocols and currently used as a baseline; SCAPI [100], one of the CRA schemes that are the most resistant to physical attacks; PASTA [80], a scalable and physical attack resistant CRA scheme, that does not require an

external *Verifier*. Unlike *mesh-like* networks, the *Internet-like* ones do not assume physical connections among nodes and each device is logically connected to any other one. To represent the connections among devices in an *Internet-like* network, SANA and SCAPI schemes can either save the status of the whole network in each single node (S1) or define only a subset of nodes as logical neighbors (S2). S1 requires a high amount of memory allocated for each node, while S2 implies an adaptation of the current protocols. A similar approach may be applied by PASTA, which should require a node to store a key for each other network device or to include a novel mechanism to trace logical neighbor status. *Thus, current CRA schemes lack a mechanism to define logical connections in a Internet-like network.* From a security perspective, *Internet-like* networks introduce more attack surfaces. The attacker can gain control of the devices from a remote location. Moreover, she can even target network infrastructure components (e.g., switches) to tamper with the packets. In general, the remote access enables the attacker to launch wider attacks that may simultaneously affect a large number of devices. PASTA is the only work that considers such scenario, but its evaluation is limited (up to 10 devices).

Security Properties To effectively defend against *injection* and *compromising* attacks, a **CRA** scheme designed for *Internet-like* networks should have three properties: (i) *neighbourhood attestation* refers to the capability of each network node to verify the integrity of its neighbours. Neighbourhood attestation permits to detect any compromised node that managed to join the network through a *compromising* attack. Neighbourhood attestation is the consequence of removing a central *Verifier* in **CRA** schemes and distributing its attestation responsibility to all the nodes. (ii) *absence detection* is the capability of a **CRA** scheme to detect whenever a device goes offline for a certain amount of time or even forever. In particular, the absence detection recognizes if a node becomes offline for a time T_a due to an *injection attack* performed through A_{bw} . (iii) *network obfuscation* refers to any strategy adopted by a **CRA** scheme to harden the network packet inspection and the consequent selective drop. This property prevents any attempt to manipulate the network traffic, which can be either performed during an *injection* or a *compromising* attack through A_{bw} or A_{sw} . Current **CRA** schemes do not provide any defence against attacks to the network infrastructure, since they assume the network is self-contained.

2.1.4 HOLA OVERVIEW

Hola is a **CRA** scheme specifically designed for *Internet-like* networks, which guarantees the security properties illustrated in section 2.1.3. Hola organizes the network devices in a ring by relying on the Chord protocol. To achieve this aim, each device needs to be equipped with specific data structures (section 2.1.4) and to manage a specific life cycle (section 2.1.4).

Hola Device Architecture. As specified in the system model, network devices are equipped with a trusted anchor (section 2.1.4). Chord algorithms, together with Hola data structures and logic, are saved in the trusted anchor of each device. Table 2.1 shows the main components of each device and details are provided below.

cert – Every node is equipped with a certificate signed with `privCAKey` and defined as a tuple containing `pubKey`, `role` and `nodeId`. The `nodeId` is an incremental integer used as the index of the node inside the **SL**. The certificate represents the device identity, is signed offline by a CA, and used to authenticate messages exchanged among devices. We accept only trusted **CAs** that are controlled by the network administrators (section 2.1.2).

SL – The **SL** keeps track of the network devices status. Each entry is a triplet defined as follows:

- `deviceStatus` (`trusted` / `offline` / `compromised`): a device is considered trusted as long as it suc-

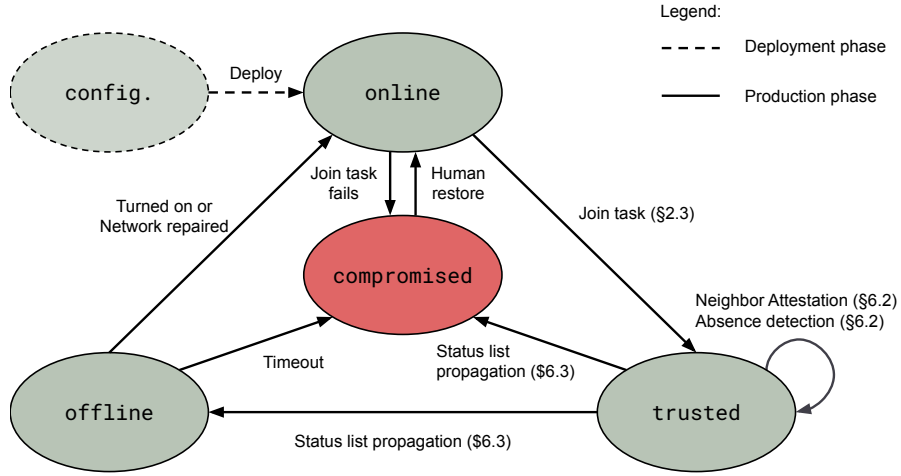


Figure 2.2: Lifecycle of a network node deployed in the HoLA CRA scheme.

ceeds the attestation from its neighbors. It eventually moves to `offline` status when it becomes inactive. Finally, a device is set as `compromised` when it stays `offline` for more than T_a or it fails the attestations.

- `exitTimestamp`: it is the exact time when a device is found to be `offline`.
- `sessionId`: it is a monotonic counter increasing every time a device enters the network. We use the `sessionId` to handle devices that temporarily go `offline` for less than T_a and that are willing to re-join the network.

We maintain a full copy of the **SL** in each device for two reasons: (R₁) it makes the HoLA scheme more robust in case of a simultaneous failure of multiple devices; (R₂) it permits any device to be aware of the whole network status, thus efficiently implementing both *single-device* and *collective* attestations (section 2.1.5). A memory-efficient way to implement the **SL** is to create a list indexed by the nodeIds.

`role` – it represents the device privilege, which could be:

- *User*: generic **IoT** device, that can add devices to the **SL**, but not remove them.
- *Admin*: dedicated devices, that can perform network maintenance and remove a device from the **SL**.

Hola Device Lifecycle.

The Hola device lifecycle is depicted in Fig. 2.2 and it involves the following states: `configuration`, `online`, `trusted`, `offline` and `compromised`.

`configuration` – Before a device is turned on, the network administrator installs the cryptographic material in the trusted anchor. In particular, the administrator provides the `cert`, the `pubKey/privKey`, and sets the `role` and the `nodeId`.

`online` – once the device is configured, it can perform the join phase following the Chord specifications (section 2.1.1). The join procedure may fail for two reasons: (1) the device has an invalid `cert` (*e.g.*, signed by an

Table 2.1: Main components of the HolA devices.

Data Structure	Short Description
<i>successors list</i>	list of the direct successors of a device (section 2.1.1)
<i>finger table</i>	list of intermediate devices in the network (section 2.1.1)
nodeId	a progressive unique number that identifies a device in the network
pubKey/privKey	keys used for issuing secure communication channels
cert	a certificate representing the device identity (section 2.1.4)
pubCAKey	the CA pubKey used for certificate validation
Status List (SL)	a structure containing the status of each device in the network (section 2.1.4)
verifySF()	function to ascertain the healthy status of a device [80, 103, 91, 92, 104, 105, 106]
role	the privilege of a device (section 2.1.4)

unauthorized CA), (2) the node has been already saved in the **SL** as compromised because of a failed neighbourhood attestation or a timeout. In both case, the device is considered compromised.

trusted – When a device is trusted, it performs the neighborhood attestation and the absence detection.

offline – Once a member device becomes unreachable, it goes offline and has to re-join the network starting from the online status.

compromised – A member device can be set as compromised by other member devices, thus becoming isolated. A compromised device can be restored only due to manual intervention from the network administrator. In this case, the device passes to online and starts the join again.

2.1.5 HOLADesign

Status List Propagation. The **SL** is a data structure containing information about the status of all network devices (*i.e.*, trusted, offline, compromised) and each network device has a local and up-to-date copy of it. Whenever a device intercepts an event that requires an update to its local **SL**, the device will then propagate the information to the network devices to make them update their local copies of the **SL**. The events that can cause an **SL** update are:

- *New device joining the network:* when a new device enters the network, it performs the Chord *join* operation. The *successor* of the new device receives information about the new entrance, updates its own **SL** and propagates the information to the network.
- *Neighbourhood attestation:* thanks to the properties of Chord, each device periodically attests its *successor*. If the device finds the *successor* either compromised or offline, it updates its own **SL** and propagates the information to the network.

- *Absence detection*: if a device has some items in its **SL** marked as `offline`, it periodically verifies if any of those devices go `online`. If a device is found `offline` for more than T_a , it is marked as `compromised` in the local **SL** and the information is propagated to the network.

The update of the local **SL** occurs through a set of priority policies. Given two **SL** entries (*i.e.*, E_1 and E_2), we assume E_1 has priority over E_2 if (i) the `deviceStatus` of E_1 is `compromised` and the `deviceStatus` of E_2 is not `compromised`; **OR** (ii) the `sessionId` of E_1 is greater than `sessionId` of E_2 . Condition (i) makes the network more conservative towards compromised devices, which might become again trusted only through the intervention of an *Admin* device. While a tampered node may be able to change its state from `compromised` to `trusted`, it would not be able to propagate it to other nodes as this particular state transition is locked by construction. Condition (ii) handles the scenario where a device turns `online` again, after being `offline`, without the network detecting its exit. The propagation of the **SL** update to the whole network refers only to the entries that need to be updated, thus reducing the amount of transferred information. A device receives the whole **SL** only when it joins the network.

Neighborhood Attestation and Absence Detection. Neighborhood attestation enables the detection of any compromised node that managed to join the network through a *compromising* attack. Absence detection permits to monitor any device going `offline` and the amount of time it stays unreachable. In *HolA*, each network device periodically performs a neighborhood attestation against its *successor*. During neighborhood attestation the *successor* receives a challenge asking to check its status through the `verifySF()` function (*i.e.*, healthy or compromised), save it into a `report`, and return it to the sender. The attestation report can depict three outcomes: *correct*, in which case nothing is done; *fail*, in which case the *successor* is marked as `compromised` in the local **SL**; *timeout*, in which case the *successor* is marked `offline`. Any update to the **SL** is eventually propagated to the neighbors. This local and up-to-date copy of the **SL** allows each network devices to perform:

- *Single-device attestation* - A device A can verify the integrity of a device B , even if B is not one of A 's neighbours by inquiring its **SL**. Previous works [92, 107] focused only on A_{sw} , while the *single-device* attestation in *HolA* detects both A_{sw} and A_{hw} .
- *Collective attestation* - an operator connects to a node with *Admin* role and looks for compromised devices in the **SL**. The operator can also manually remove the compromised nodes from the **SL**. We remark that the `node role` is part of the `cert`. Thus, an adversary cannot impersonate an *Admin* node, unless she breaks the CA signature or steals a node with *Admin* role.

Network Obfuscation. In our threat model, we assume that the attacker may perform statistical analysis over the exchanged packets (section 2.1.2). For instance, she may detect and stop those packets belonging to the **SL** propagation, thus stopping the updates on compromised nodes. Since we assume the devices employ secure cryptography primitives, we exclude man-in-the-middle attacks. However, the adversary can still observe the sender IP, the receiver IP, and the packet size. Previous works showed that this information is enough to de-anonymize the packets [108, 109]. To mitigate this issue, we harden the packet analysis by employing network obfuscation strategies. In particular, we took inspiration from two techniques used in the mix-networks [110]: (i) all the exchanged messages have the same size by design, and (ii) any device sends extra random packets to a fixed set of devices. These techniques avoid any intermediate device to distinguish between **SL** propagation, neighborhood

attestation, or Chord routines. In addition, this disrupts a frequency analysis to discover the *successors* of a device [111]. We analyze the packet size and quantify the network overhead in (section 2.1.6) and (section 2.1.6), respectively.

2.1.6 HOLA EVALUATION

Experimental setup. We used three setups: Raspberry Pi 0 [112], Raspberry Pi 3 [113], and a simulated network on Omnet++ [89]. We used a Raspberry Pi 0 [112] as a constrained environment to estimate the cost of cryptographic operations used by HolA. A Raspberry Pi 0 mounts a 1GHz single-core CPU with 512MB RAM. Each received message requires three cryptographic operations to be performed: (i) authentication, (ii) key negotiation, and (iii) decryption. The authentication mechanism uses an RSA schema with a 2048bits long key. The certificate verification required on average 0.589ms (0.02ms std). The key generation phase produces an AES 256B long key using a Diffie-Hellman exponentiation. This operation required on average 5.92ms (0.0522ms std). The decryption mechanism uses an AES-GCM [114] with a 256B long key. It required on average 0.0833ms (0.126ms std). We used a network of 5 Raspberry Pi 3 [113] to prove the feasibility of HolA on real devices mounting ARM TrustZone [115]. We developed the prototype on top of OP-TEE [116] by using the C language. We implemented network communication using two TCP sockets opened in the *untrusted zone*. We used Omnet++ to simulate an IoT TCP/IP network with up to 1M devices. We set a delay of 10ms to simulate message processing based on the measurements on the Raspberry Pi 0. We set the communication rate to 250Kbps based on the defined data-rate of the 6LoWPAN specifications [86]. We compared the results of our experiments with state-of-the-art solutions [102, 80, 100]. Each work has proposed its own experiments and units of measure. We therefore tried to proposed a thorough comparison with the information available.

HolA Resiliency. HolA requires that the underlying ring is preserved to automatically repair itself in case of disrupted nodes. Therefore, the overall security of our protocol is strictly related to this property. We can adjust the resilience of our network by tuning the *successors list* length (*SLEN*). In particular, *SLEN* must be longer than the longest sequence of consecutive disrupted nodes.* Fig. 2.3a shows the minimum *successors list* length required for preserving the ring structure in case of random nodes disappearing. For the sake of this evaluation, we consider the nodes disappearing either for failure or attack, while we distinguish these cases in (section 2.1.6). Fig. 2.3a shows a logarithmic trend, therefore, the cost for a single node to maintain the structure scales *w.r.t.* the size of the network. In the remainder of this section, we will consider the 50% average values as a reference (*i.e.*, 4 for 10 nodes, 7 for 100 nodes, 10 for 1K nodes, 14 for 10K nodes, 17 for 100K nodes and 20 for 1M nodes). Our results show that HolA is in line with [100] by tolerating up to $n/2$ outages in the network. In addition, HolA can adjust its resiliency by increasing the *successors list* length, thus introducing the unique possibility to trade-off between resiliency and memory overhead.

HolA Security Properties The objective of HolA is to maintain a healthy status of the network (section 2.1.2). This is achieved by periodically challenging the nodes and broadcasting the updates. We show the correctness of this approach by formal demonstration.

Definition 3. *A node is compromised if it does not pass the neighbourhood attestation or if it is found offline by the absence detection and it stays offline for more than a time T_a (section 2.1.5).*

*With consecutive nodes, we mean nodes with a consecutive position in the Chord ring.

Definition 3 derives directly from the threat model considerations (section 2.1.2) and HolA design (section 2.1.5).

Definition 4. *A network attestation protocol is secure if it prevents compromised nodes to operate in the network.*

Definition 4 indicates that each node belonging to the network must receive *neighbourhood attestation* and that each offline node must be detected by *absence detection*. To avoid compromised nodes to rejoin the network, HolA must additionally broadcast the information to all the devices inside the network (section 2.1.5).

Theorem 1. *Neighbourhood attestation guarantees continuous check of nodes.*

Proof sketch. In HolA, each node receives a challenge by its predecessor in the ring (section 2.1.5). Given the resiliency property (section 2.1.6), the ring always remains intact during HolA operations. Therefore, a node participating to the protocol will always have a predecessor.

In HolA, a node continuously receives challenges to check its status and eventually detect a *compromising attack* carried out through A_{sw} or A_{bw} (section 2.1.3). Therefore, it is crucial that any node is constantly verified by the other nodes; this time may vary due to the presence of offline nodes. For instance, a node immediately after a disruption needs to wait for the protocol to repair the ring before receiving its next challenge. We deepen this aspect in (section 2.1.6).

Theorem 2. *Absence detection guarantees the detection of offline nodes.*

Proof sketch. Each node periodically sends a challenge to its successor and expects a response. In case the communication times out, the node flags the successor as offline in the status list, removes it from the successors list and propagates this information. The sequence of operations is repeated until the node finds the first online device in the successors list. The resiliency property (section 2.1.6) assures that at last one node will eventually find an online device. Each node participating to the protocol performs this task. Hence, using again the resiliency property, there will always be a node detecting and propagating the information about its offline successors.

Since each node of the network is continuously proved, HolA avoids *injection attacks* (section 2.1.3).

Theorem 3. *Status list propagation reaches all online devices.*

Proof sketch. Each node propagates the information to its successors list. Given the resiliency property (section 2.1.6), the ring always remains intact during HolA operations. Hence, the information will be routed around the circle and will eventually reach all the nodes.

Without losing in generality, we considered a limit case in which a node has an empty *finger list*, thus only relying on its *successor list*. In reality, the *finger table* usually has some entries that can be used to improve the broadcast speed. We measure the impact of the *finger table* in (section 2.1.6). Moreover, we distinguish between a random attacker and a selective attacker. Section 2.1.6 treats the first case demonstrating the robustness of the process with different percentages of compromised nodes. For the second case, a selective attacker needs to physically compromise *SLEN* consecutive nodes in the ring to block the propagation around the ring. Since we include network obfuscation techniques (section 2.1.5), an adversary cannot exactly locate the *successors*. To successfully remove all the successors, an adversary must control all the devices contacted by a target node and those contacted by its *successor list*. In practice, in a network of 1M of device, considering a *SLEN* of 20, and a finger list of 24 devices (more detail in Fig. 2.3e), an attacker must control 24^{20} devices (around 4×10^{27}) at the same time to block

the **SL** propagation. To summarize, the network obfuscation enables a secure **SL** propagation to resist against a *compromising attack*.

Time Delay for Neighbourhood Attestation. We evaluated the time delay that neighbourhood attestations could suffer due to network disruptions. We experimented with 6 different populations presenting 4 different percentages of random offline nodes (from 0 to 50%). Fig. 2.3b shows a logarithmic trend, therefore, the time period scales *w.r.t.* network size. The largest period is reached with a population of 1M device, where the maximum expected period is 1.3s. Despite major disruptions, the neighbourhood attestation procedure demonstrates optimal performance in terms of scalability and availability.

SL Propagation Performance. Theorem 3 demonstrates the correctness of HolA propagation. However, the overall performance of the naive implementation (*i.e.*, propagating only to *successors list* nodes) are linear *w.r.t.* the network size. To improve it, we experimented with a more efficient propagation where the information is additionally sent to one entry in the *finger table*. From the propagation perspective, this is equivalent to spread the information using a binary tree topology. We measured the time and number of messages required for the propagation to reach every node in the network. We experimented with 6 different populations and 4 different percentages of random offline nodes. For the *successors list* lengths, we considered the suggested values in (section 2.1.6). Fig. 2.3c shows the average propagation time required for HolA to propagate an information to the whole network. The propagation time is almost logarithmic, demonstrating the scalability of the process *w.r.t.* the network size and that the process is slightly affected by the rate of offline nodes. The propagation time ranges from 8s to 10s with a population of 1M devices. In terms of messages, Fig. 2.3d shows the average number of messages required increases linearly with the online population and the number of messages sent by each node. In fact, each online node sends out an exact number of messages (all the successors plus a finger). As an example, for a population of 1M devices where 50% of them are offline we measured exactly 10,500,000 messages $(50\% \times 1M) \times (20 + 1)$, where 20 is the *successors list* length and 1 the additional finger. This simplifies the estimate of the burden introduced by extra random packets (section 2.1.5). In fact, the number of random packets sent proportionally affects the global number of packets exchanged. In other words, r extra random packets per each node would bring an increase of r -times in the messages. HolA guarantees an efficient network propagation that scales quasi-logarithmically *w.r.t.* the network size. A complete execution of the protocol comprising both the attestation and the propagation takes an average time period of around 12s (2.1.6). This is a major improvement *w.r.t.* to previous works, where the elapsed time could be as long as 71.7s [80] and 1421s [102]. The time required by PASTA [80] highly depends on the network's state, ranging from 3s to 71.7s while our results are more stable.

Memory Consumption. The *successor list* size (*SLEN*) is a constant value to be set before deploying the network. Each entry in the *successor list* stores an IP address (4B) together with the *pubKey* of the related node (128B), for a total of 132B. Hence, a complete *successor list* requires $(132 \times SLEN)$ B. Every entry of the *finger table* requires 132B as for the *successor list* (4B for the IP plus 128B for the *pubKey*). The total number of entries in the *finger table* depends on the network size. Fig. 2.3e shows the maximum number of fingers of the *finger table* with different network populations (up to 1M devices). The plot shows a logarithmic trend, suggesting that the *fingers table* scales *w.r.t.* the network size. In particular, a *finger table* contains $132 \times \log_2(n)$ B. Each **SL** entry stores a *deviceStatus* (1B), an

exitTimestamp (8B) and a *sessionId* (1B), totalling 10B. The **SL** must contain all the devices in the network, hence the overall memory overhead is $(10 \times n)$ B. We introduced a cache for storing the *pubKey* and *nodeId* of other devices and speed up the communications. In particular, we tune the cache to contain as many entries as

the expected number of nodes to contact (Fig. 2.3f). The graph indicates a logarithmic growth *w.r.t.* the global population of the network. This suggests that a cache could reduce the burden of certificate verification without imposing too much overhead on the memory. Each entry in cache occupies 130B (128B for the pubKey and 2B for the nodeId), setting the cache memory to $(130 \times \log_2(n))$ B. The overall memory used for data structures is $10 \times n + 262 \times \log_2(n) + 132 \times SLEN$. The other decentralized autonomous network (PASTA [80]) claims an overall memory cost of at most $(78,140 + |token| \times 1,280)$ B (around 700Kb). In the same scenario (*i.e.*, 10K), HolA has an overhead of at most $(103,668 + 132 \times SLEN)$ B. Considering 14 as $SLEN$ (section 2.1.6), the size becomes 105,516B (around 100Kb), that is seven times less overhead *w.r.t.* PASTA.

Communication Overhead. We measure the communication overhead in terms of message size. In our implementation, the messages contains a header of 274B, where the majority part is dedicated to a *certificate* (256B) and the rest is for the HolA internal working. The payload of a message depends in which phase HolA is operating: (i) *join phase*, and (ii) *operational phase*. During the *join phase*, the largest message sent has size $(10 \times n + 132 \times SLEN + 274)$ B and it is sent once. The *operational phase*, instead, comprises all the tasks executed by a node during its permanence inside the ring. Since we adopt network obfuscation techniques, all the *operational* messages has the same length, which is of $(264 \times (SLEN + 1))$ B. In case of a network of 10K devices and a resiliency rate of 50% (section 2.1.6), the HolA *operational* messages have a weight of 3,960B. SANA [102] has a communication costs in the same order of magnitude of HolA. The same can be said for SCAP [100] that claims an average cost of 1,314B for a network of 10K devices.

2.1.7 RELATED WORKS

CRA on spanning tree topology. SEDA [117], SANA [102], and LISA [118] are the first **CRA** schemes. They initiate a spanning tree topology over the network and use it for distributing the burden of computation among all the devices. A limitation regarding these approaches is the static topology assumption that forces the devices to not disrupt the tree during the protocol execution. Moreover, they do not consider hardware attackers [117, 102, 118], they provide coarse-grained results [117] or expensive aggregation methods [102] and they propose inefficient secret keys management [117, 118].

Physical attacks detection. Due to the increasing size of the networks, researchers investigated mechanisms to detect *physical* adversaries that may capture the devices. These works assume a *physical* adversary removes a device from the network for a non-negligible amount of time. DARPA [78] and SCAP [100] are the main works in this direction. They both rely on an heartbeat token exchanged between neighbors that permits an external *Verifier* to detect devices' absence. These works act as overlays placed above existing solutions (SEDA [117], SANA [102]). Thus, they inherit both the complexity and the limitations of the underlying attestation scheme.

CRA for highly dynamic swarms. To tackle the static topology issue of first **CRA** schemes, more recent ones proposed a scheme that incrementally creates a complete snapshot of the network status. SALAD [107] and PADS [119] achieve this goal through a shared structure that contains the status of all the devices, and an external *Verifier* that needs to retrieve the structure from a random device.

Autonomous networks. A new branch of **CRA** works proposes autonomous networks that do not rely on an external *Verifier* to maintain their healthy status. DIAT [92] assumes a *mesh-like* connection and focuses on *software* adversaries and run-time RA (*i.e.*, they validate runtime device status). US-AID [81] can detect both *software* tampering and device disconnections. PASTA [80] handles both *software* and *physical* adversaries by relying on a

periodical generation of tokens that attests the integrity of all the devices that participated in its generation. These autonomous **CRA** schemes focus on *mesh-like* networks where a device can only connect with its physical neighbours. Hence, they do not provide support for those environments in which a device can potentially connect with all the others.

2.1.8 DISCUSSION

Certificate revocation/expiration. In our design, we ruled out the certification revocation and expiration. This may let adversaries forge fake certificates and allow malicious devices to join the network. We can overcome this issue with the introduction of probabilistic filters, as described in previous works [120].

False positive. A device is considered compromised through A_{hw} , if it is offline for more than T_d . However, this does not mean a device has been actually under attack. This is also considered as an open problem in previous works [80, 81, 100]. In HolA, we mitigate this issue by relying on *Admin* devices that can manually control the network status and restore outage devices.

Devices loosely synchronized. As assumed also in previous works [80, 81, 100], the network devices require some clock synchronization strategy to detect a A_{hw} . We aim at overcoming this limitation by storing the relative time at which a device becomes offline, instead of the absolute timestamp. However, this introduces other synchronization challenges, such as considering random network propagation delays. We plan to investigate new solutions for this issue in future versions of HolA.

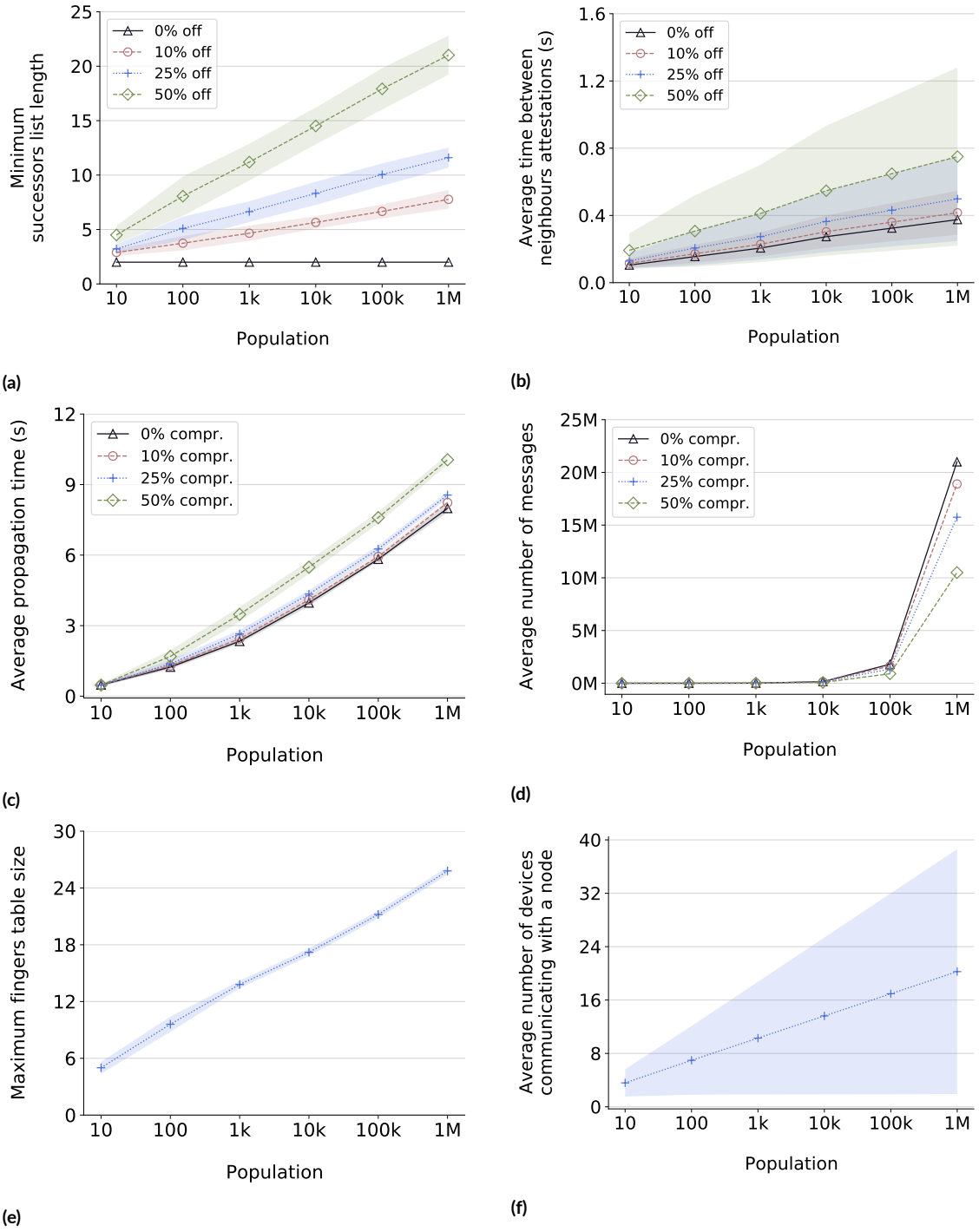


Figure 2.3: Minimum successors list length for preserving the ring structure (a), average time between two consecutive challenges (b), average time required for a broadcast (c), average number of messages for broadcasting (d), maximum fingers list size (e), average number of devices communicating with a node (f). The lines represent offline nodes (a,b) and compromised nodes (c,d). Shaded areas are the standard deviation.

2.2 X-LOCK: A SECURE XOR-BASED FUZZY EXTRACTOR FOR RESOURCE-CONSTRAINED DEVICES

Within the domain of **IoT**, the presence of resource-constrained devices poses significant obstacles in ensuring the development of robust privacy and security mechanisms. While numerous cryptographic algorithms can be customized to address these challenges, their effectiveness hinges on the availability of securely maintained keys. Presently, prevalent practices involve the storage of digital keys in a **NVM**, typically situated externally to the computing platform. However, the practical realization of secure digital key storage has emerged as a formidable endeavor, primarily attributed to technical limitations or cost-related considerations [121].

In this context, **PUF** [122] arise as a promising alternative solution. **PUF** offer a viable replacement for **NVM**-based keys and offer several notable advantages, including cost-effectiveness, inherent uniqueness, and heightened resistance against various attacks [123]. **PUF** derive confidential information from inherent process variations, similar to distinctive device fingerprints [124]. The replication of identical **PUF** instances becomes then practically unattainable, even when originating from the same manufacturer. Nevertheless, **PUF** rely on physical circuit properties for generating responses, making them susceptible to factors such as thermal noise and environmental conditions. Consequently, achieving reliable replication of **PUF** responses poses a significant challenge [121].

The notion of fuzzy extractors [125] emerged as a highly regarded approach for addressing key management issues associated with error-prone data. A fuzzy extractor offers the ability to extract an identical random string from a noisy source without the need to store the string itself. The general construction comprises two algorithms. The generation algorithm *Gen* takes an initial string S_{read} of the noisy source as input and produces a string K along with a helper data H . Subsequently, the reproduction algorithm *Rep* leverages the helper data H to reproduce the string K from a second string S'_{read} of the same source, given that the distance between S_{read} and S'_{read} is sufficiently small. The correctness property of a fuzzy extractor ensures that the reproduction process yields the same string K when the fuzzy data is generated from the same source. Additionally, the security aspect of a fuzzy extractor guarantees that the helper data H does not divulge any information about the original fuzzy data.

Previous works employed secure sketches and randomness extractors as core components for their constructions [126, 127, 128]. A secure sketch is an information reconciliation protocol that enables the recovery of the original S_{read} from a received S'_{read} when they are sufficiently close. Subsequently, a random string is extracted from S_{read} using a randomness extractor. However, it has been observed that secure sketches leak information through the helper data, leading to a loss of security [129]. Consequently, the construction of a fuzzy extractor-based on secure sketches necessitates the utilization of high entropy source data.

Recently, a novel class of solutions introduced a novel approach to construct fuzzy extractors [129, 130, 131]. These solutions employ S_{read} to encrypt a random string in a manner that allows for decryption with knowledge of a closely related string S'_{read} . Fuller et al. [129] presented a computational fuzzy extractor based on the **LWE** problem. Their approach effectively mitigates information leakage from the helper data by concealing secret randomness within it. However, this construction exhibits notable inefficiency and only tolerates sub-linear errors. Additionally, it lacks guarantees of reusability and robustness. Canetti et al. [130] employed digital lockers as their cryptographic primitive. Their solution is applicable even with low-entropy sources and ensures reusability. However, their sample-then-lock technique necessitates excessive storage space for the helper values. Woo et al. [131] proposed a solution based on non-linear **LWE**, which offers memory efficiency while simultaneously guarantee-

ing both reusability and robustness. Nevertheless, their solution is computationally demanding and may not be suitable for restricted environments.

This section presents X-Lock, a novel design for a secure and cost-effective computational fuzzy extractor. Similar to [129, 130, 131], X-Lock utilizes the noisy strings obtained from a fuzzy source to protect a random string of bits. X-Lock encrypts each bit from the random string by XOR-ing it multiple times with a subset of bits from the noisy string. Through this approach, X-Lock offers reusability and robustness while inherently addressing bias and correlation issues within the fuzzy source. To substantiate these assertions, we conducted a comprehensive theoretical analysis encompassing both the security and implementation aspects of our model. Furthermore, we conducted practical experiments and comparisons with existing state-of-the-art approaches to demonstrate the superiority of our proposal.

The empirical results demonstrate the optimal performance of our proposed model. With a key size of 128 bits, our algorithm uses $\approx 5KB$ to store 9 keys. Furthermore, the execution time for the *Rep* procedure amounts to 1.32ms. In contrast, Canetti et al. [130] necessitates 297GB and $> 1min$ computational time, while Woo et al. [131] utilizes $\approx 5KB$ and operates at a speed of 7.0s. Overall, our model achieves comparable memory cost with the state-of-the-art solutions while outperforming them by being 3 orders of magnitude faster.

2.2.1 RELATED WORKS

Ever since the pioneering work of Dodis et al. [125], the concept of fuzzy extractor has emerged as a prominent solution for managing keys derived from noisy sources. Most fuzzy extractors adopt the sketch-then-extract paradigm, employing secure sketches and randomness extractors as fundamental components [126, 127, 128]. However, secure sketches entail a minor data leakage from the helper data, leading to compromised security [129]. Consequently, the utilization of a secure sketch-based fuzzy extractor necessitates high-entropy source data.

The notion of reusability, as formalized by Boyen [126], pertains to the security of multiple pairs of extracted strings and associated helper data, even when correlated source data is exposed to an adversary. The work showcased that achieving information-theoretic reusability requires a significant reduction in security, thereby implying an inherent trade-off. Related again to fuzzy extractors security, Boyen et al. [132] introduced the notion of a robust fuzzy extractor to safeguard against malicious modifications of the helper data, ensuring the detection of any such alterations. Both these security criteria must be satisfied to ensure the viability of fuzzy extractors as secure authentication methods in real-life scenarios.

In a departure from secure sketch-based approaches, Fuller et al. [129] proposed a computational fuzzy extractor based on the **LWE** problem. Their methodology employs noisy strings to encrypt a random string in such a way that it can be decrypted with the knowledge of another closely-related string. By concealing secret randomness within the helper data instead of extracting it from the noisy string, their approach mitigates data leakage concerns. Nonetheless, their construction exhibits significant inefficiency and can only tolerate sub-linear errors, while lacking reusability. Apon et al. [133] improved upon Fuller’s construction to fulfill the reusability requirement by utilizing either a random oracle or a lattice-based symmetric encryption technique. However, their solution remains unable to overcome the limitations of Fuller’s construction, allowing only a logarithmic fraction of error tolerance. Another proposal by Wen et al. [127] presented a reusable fuzzy extractor based on the **LWE** problem, resilient to linear fractions of errors. Nevertheless, their scheme relies on a secure sketch and consequently results in the leakage of sensitive data.

Addressing the limitations of prior works, Canetti et al. [130] introduced a fuzzy extractor construction employing inputs from low-entropy distributions, leveraging the concept of digital lockers as a symmetric key cryptographic primitive. Their strategy involves sampling multiple partial strings from a noisy input string, hashing them independently, and then locking individual secret keys with each hashed partial string. Correctness ensues if there is a high likelihood of successfully recovering at least one hashed subset of data upon a second measurement. To achieve this, the scheme necessitates a large helper data size for storing the hashed values and employs an inefficient reproduction algorithm. To address this concern, Cheon et al. [134] made adjustments to the scheme, reducing the size of the helper data. However, this modification results in an increased computational cost for the reproduction algorithm.

A recent contribution by Woo et al. [131] presents a novel computational fuzzy extractor that does not rely on a secure sketch or digital lockers. Their construction offers security under the non-linear **LWE** problem and encodes the extracted key using two cryptographic primitives: error correcting codes (ECCs) and the EMBLEM encoding method [135]. This innovative approach achieves both robustness and reusability while tolerating linear errors. However, it assumes that the source data is drawn from a uniform distribution and, although more efficient than previous works, it still entails a non-negligible computational effort that may limit its adoption on low-end devices.

In contrast with previous works, X-Lock efficiently satisfies all the security properties required from a secure fuzzy extractor. The protocol only uses XOR operations in its procedure, thus cutting the computational complexity. Additionally, the XOR-ation primitive inherently introduces a mitigation for both source bias and correlation issues. It is noteworthy that none of the previous works adequately addresses the bias and correlation issues. In real-world setups, fuzzy sources rarely produce perfectly random outputs and may exhibit biases towards a specific value, as well as correlations among bits. While Canetti’s construction [130] implicitly addresses the correlation problem, the bias issue remains unaddressed in the existing literature.

2.2.2 BACKGROUND

General notation. We use lowercase letters to denote single values. We use capital letters to denote groups of values that are either organized in sets or strings. With $\{\cdot\}^x$ we denote a group of cardinality x that contains values from those described inside the braces. We denote with $|X|$ the cardinality of the group X . We define $||X||$ as the Hamming weight of X , i.e. the number of its non-zero values. The notation $Pr[\cdot]$ expresses the probability of the event described inside the square brackets. We denote the XOR logical operation with $\oplus$. When applied to sequences of bits, $\oplus$ is intended to perform a bit-wise XOR. We introduce the XOR-ation operator $\bigoplus_{i=0}^I$, performing the XOR operation iteratively over a sequence of bits denoted by the indexes $0 \leq i \leq I$. Note that the order of the indexes does not change the final result.

PUF characterization. PUF leverage the inherent random variations introduced during the silicon manufacturing process to generate secret keys on the fly [122]. These variations, akin to unique fingerprints, serve as a cost-effective source of randomness. Conceptually, a PUF can be viewed as a function where a specific binary input (i.e., a challenge) elicits distinct binary outputs (i.e., responses) specific to each individual PUF instance. However, outputs obtained from the same PUF circuit may exhibit variations due to factors such as thermal noise and environmental conditions. Consequently, we conceptualize a PUF circuit as a fuzzy source F that provides a string of bits S_{read} upon request. When comparing two such sequences from F they can exhibit different values in certain

positions. We formalize this aspect with the relative error $e_{rel} = \frac{||S_{read} \oplus S'_{read}||}{|S_{read}|}$, that is the ratio of bits in S_{read} that differ from S'_{read} . Together with S_{read} , we introduce the concept of preferred string S_{pref} , which is obtained by identifying the value appearing most frequently (*i.e.*, more than 50% of the times) for each bit within the string. We calculate it by using a statistically significant number of S_{read} (*e.g.*, more than 100 strings). Similarly to e_{rel} , we define the absolute error e_{abs} as the ratio of bits in S_{read} that differ from S_{pref} . Notably, the two errors are linked by the inequality $e_{abs} \leq e_{rel} \leq 2 \cdot e_{abs}$. It is crucial to underscore that when setting up the fuzzy extractor, using the raw S_{read} instead of S_{pref} is likely to introduce additional errors into the process, ultimately increasing the likelihood of recovering an incorrect key. Hence, in the context of this section, we employ S_{pref} as our reference string, and consequently, we consider e_{abs} to be the error rate associated with F .

Fuzzy extractors. Fuzzy extractors [125] are cryptographic constructions designed to derive reliable and uniformly distributed cryptographic keys from sources prone to noise or errors. This process involves two probabilistic procedures. The *Gen* procedure takes a string S_{read} and produces a random string K along with a public helper string H . If another string S'_{read} is sufficiently close to S_{read} (*i.e.*, $||S_{read} \oplus S'_{read}|| \leq t$ for a small t), the *Rep* procedure can utilize the helper string H to reproduce the original random string K from S'_{read} .

In this work, we focus on computational fuzzy extractors that consider the scenario where potential attackers possess knowledge about the noise distribution and have control over the errors. To formalize this concept, we define a metric space as a finite set $\mathcal{M}$ equipped with a distance function $dis : \mathcal{M} \times \mathcal{M} \leftarrow \mathbb{R}^+ = [0, \infty)$ that satisfies the identity property ($dis(x, y) = 0$ if and only if $x = y$), the symmetry property ($dis(x, y) = dis(y, x)$), and the triangle inequality ($dis(x, z) \leq dis(x, y) + dis(y, z)$). The statistical distance $SD(X, Y)$ is defined as $\frac{1}{2} \sum_x (Pr[X = x] - Pr[Y = y])$.

Definition 2.2.1 (Computational fuzzy extractor). *Given a metric space $(\mathcal{M}, dis)$, let $\mathcal{F}$ be a family of probability distributions over $\mathcal{M}$. A pair of randomized procedures (Gen, Rep) is an $(\mathcal{M}, \mathcal{F}, k, t, \epsilon)$ -computational fuzzy extractor with error δ if Gen and Rep satisfy the following properties:*

- *Gen on input $S_{read} \in \mathcal{M}$ outputs $K \in \{0, 1\}^{|K|}$ and a helper string $H \in \{0, 1\}^*$.*
- *Rep takes $S'_{read} \in \mathcal{M}$ and $H \in \{0, 1\}^*$ as inputs.*
- **Correctness.** *If $dis(S_{read}, S'_{read}) \leq t$ and $(K, H) \leftarrow Gen(S_{read})$, then $Pr[Rep(S'_{read}, H) = K] \geq 1 - \delta$, where the probability is over the randomness of (Gen, Rep) .*
- **Security.** *For any $F \in \mathcal{F}$, K is pseudo-random conditioned on H , that is if $(K, H) \leftarrow Gen(S_{read})$ then $SD((K, H), (K', H)) \leq \epsilon$.*

The correctness property guarantees that the fuzzy extractor can accurately retrieve the protected data when provided with an input that is sufficiently close to the original. The security property ensures that the output of the fuzzy extractor does not reveal any specific information about the underlying noisy distribution.

To utilize a computational fuzzy extractor as an authentication mechanism, it must possess both reusability and robustness. Reusability [126] is related to the case where several pairs of extracted string and related helper data issued from correlated data are revealed to an adversary.

Definition 2.2.2 (Reusability). *Let $\mathcal{F}$ be a family of distributions over $\mathcal{M}$. Let $(F^1, F^2, \dots, F^\rho)$ be ρ correlated random variables such that $\forall i \in \rho : S^i \in \mathcal{F}$. Let (Gen, Rep) be a $(\mathcal{M}, \mathcal{F}, k, t, \epsilon)$ -computational fuzzy extractor*

with error δ . Let D be a distinguisher that outputs 0 when it believes that the input was randomly produced and 1 when it believes it was produced by (Gen, Rep) . Define the following game $\forall j = 1, \dots, \rho$:

- **Sampling.** The challenger samples $S_{\text{read}}^j \leftarrow F$ and $K \leftarrow \{0, 1\}^{|K|}$.
- **Generation.** The challenger computes $(K^j, H^j) \leftarrow \text{Gen}(S_{\text{read}}^j)$.
- **Distinguishing.** The advantage of D is quantifiable as $\text{Adv}(D) = \Pr[D(K^1, \dots, K^{j-1}, K^j, \dots, K^\rho, H^1, \dots, H^\rho) = 1] - \Pr[D(K^1, \dots, K^{j-1}, K^{j+1}, \dots, K^\rho, H^1, \dots, H^\rho) = 1]$.

(Gen, Rep) is (ρ, σ) -reusable if $\forall D \in \mathcal{D}$ and $\forall j = 1, \dots, \rho$, the advantage is at most σ .

As underlined in Canetti et al. [130], the key aspect is that the family of distributions $\mathcal{F}$ is arbitrarily correlated, meaning no assumption is made on the correlation between the distributions.

Robustness [132] addresses the scenario where an adversary modifies the helper data H before it is given to the user. A robust fuzzy extractor ensures that any changes made to H by an adversary will be detected.

Definition 2.2.3 (Robustness). Let $\mathcal{F}$ be a family of distributions over $\mathcal{M}$. Let $\perp$ denote that Rep detected a tampered output. Let (Gen, Rep) be a $(\mathcal{M}, \mathcal{F}, k, t, \epsilon)$ -computational fuzzy extractor with error δ . Let $(K, H) \leftarrow \text{Gen}(S_{\text{read}})$, with S_{read} output of F . Let A be an adversary, and $H' \leftarrow A(K, H)$ with $H \neq H'$. Then (Gen, Rep) is a τ -robust fuzzy extractor if $\Pr[\text{Rep}(S'_{\text{read}}, H') \neq \perp] \leq \tau$.

Bias and correlation in fuzzy sources. In real-world scenarios, fuzzy sources often exhibit non-uniform distributions in their output bit strings. These deviations from uniformity can arise from two main factors: bias and correlation among the physical components of the source. When bias is present [136], it means that either 0-bits or 1-bits occur more frequently than the other. We quantify bias by the parameter p_{bias} , with $p_{\text{bias}} \cdot l$ being the expected number of 1-bits in a l -bit response. An unbiased source would have $p_{\text{bias}} = 0.5$, indicating an equal probability for 0-bits and 1-bits. Correlation, on the other hand [137], refers to the lack of independence among the values of the bits generated by the noisy source. This lack of independence arises due to physical dependencies, such as cross-talk noise among SRAM cells [138], where the value of one bit can be influenced by neighboring bits in the physical medium. To precisely define the concept of correlation, we introduce the notion of correlation classes of bits.

Definition 2.2.4 (Correlation class of a bit). Let $F \in \mathcal{F}$ be a fuzzy source drawn from a family of noisy sources $\mathcal{F}$. Let $\sim$ denote the equivalence relation. Let S_{pref} be the preferred string of F . Then, the correlation class $[b]$ of bit b is the set $\{b' \in S_{\text{pref}} : b' \sim b\}$ of elements that are equivalent to b .

It is important to note that the correlation classes are disjoint, meaning that a bit can only belong to a single class. The total number of correlation classes corresponds to the maximum number of independent values that can be extracted from a particular source. In this work, we assume that the correlation classes have approximately the same size, meaning that each independent value has an equal probability of being selected in a random draw. To quantify the level of correlation, we introduce the correlation factor ϕ .

Definition 2.2.5 (Correlation factor). Let $F \in \mathcal{F}$ be a fuzzy source drawn from a family of noisy sources $\mathcal{F}$. Let S_{pref} be the preferred string of F . Let $[S_{\text{pref}}]$ denote the set of all correlation classes in S_{pref} and $||[S_{\text{pref}}]||$ the total number of correlation classes in S_{pref} . Then, the correlation factor is defined as $\phi = 1 - \frac{||[S_{\text{pref}}]||}{|S_{\text{pref}}|}$.

The correlation factor provides an immediate measure of dependency between the source bits. For example, a correlation factor of $\varphi = 0.75$ indicates that we expect three dependant values for every four bits in the source.

Threat model. In our scenario, we consider an adversary who has complete control over the communication channels [101]. The adversary has access to the appliance that executes the algorithm and stores its data. However, we explicitly exclude the possibility of extensive physical attacks or invasive side-channel attacks by the adversary. These types of attacks would require unrestricted access to the device for extended periods of time, which would raise suspicion and allow for their detection. Furthermore, we assume that the adversary can read data from standard non-volatile memory and modify it. However, the **PUF** in the system possesses a tamper-evidence property. If the adversary attempts to learn the secret stored in it, the behavior of the **PUF** will change significantly or be destroyed, thereby indicating the tampering. Lastly, we assume that all algorithms related to our protocol are public, but implemented in a way that prevents modification. This assumption ensures that the adversary cannot tamper with or modify the algorithms to their advantage.

To conclude, we provide a summary of the notation used throughout the section.

Table 2.2: Notation summary.

Symbol	Meaning
$ X $	Cardinality of group X
$ X $	Hamming weight of group X
$\bigoplus_{y \in Y}$	XOR-ation over the set of indexes Y
F	Fuzzy source
S_{read}, S_{pref}	Random string and preferred string of a fuzzy source
e_{rel}, e_{abs}	Relative error and absolute error
Gen, Rep	Procedures involved in a fuzzy extractor
K, H	Secret key and helper data generated by Gen
p_{bias}	Number of 1s in a string over its total number of bits
φ	Correlation factor of a fuzzy source
B	Poll of random bits
C	Bits involved in the XOR-ation
L	Bit locker, group of locks relative to the same bit
K_{pre}	Final values before obtaining K
V	Vault
n	nonce
R	Set of random indices of original bits constituting a key
T	Authentication token
$\perp$	None value

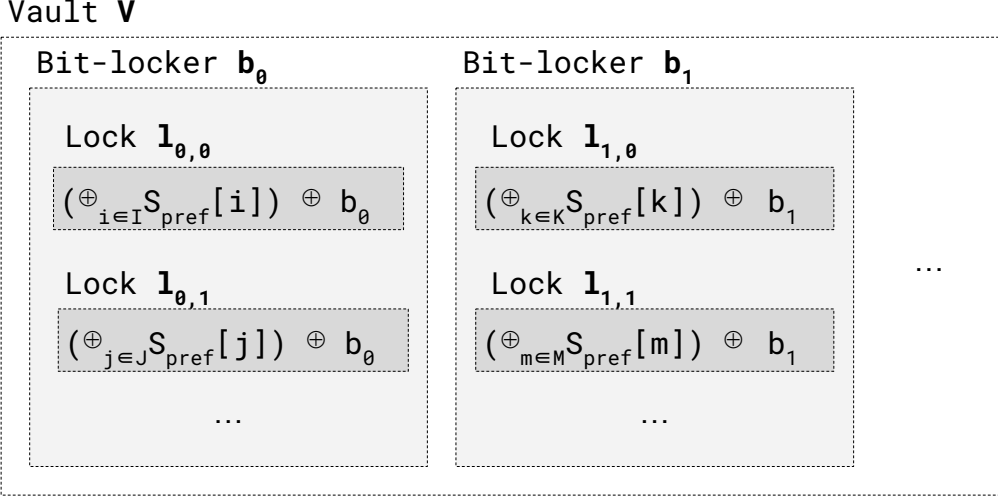


Figure 2.4: Composition of the vault V . It is a collection of bit-lockers, each referring to a specific bit $b \in B$. A bit-locker contains multiple locks that XOR the bit of reference with the XOR-ation of a random subset of bits from the source F .

2.2.3 X-LOCK: CONSTRUCTION DETAILS

We consider a noisy source F that outputs a bit-string S_{read} of length $|S_{read}|$ upon request. We denote S_{pref} as the preferred state of F and e_{rel}, e_{abs} to be respectively its relative and absolute errors. Section 2.2.2 provides a thorough description of these quantities. Similarly to existing works [129, 130, 131], our algorithm shares a common thread in leveraging the outputs generated by F to encrypt a random string of bits B . However, we significantly differentiate from the previous proposals as we employ S_{pref} to encrypt a large pool of bits B by means of the XOR-ation operator. We then decrypt and combine random subsets of B to generate multiple secret keys. The core element of the algorithm is the vault V resulting from the encryption of B . We provide reference of the structure of V in Figure 2.4. It consists of a sequence of bit-lockers, which individually guard the encryption of a single bit $b \in B$. Each bit-locker is a collection of locks L , where each lock is the XOR of a distinct b with the XOR-ation of a random subset of bits from S_{pref} . To restore the bit value b is then sufficient to retrieve a S_{read} from F and perform the XOR-ation on the lock value with same subset of bits. In section 2.2.4 we show that this construction indeed forms a valid computational fuzzy extractor.

To perform the initial encryption of B into V , our algorithm introduces an *Init* procedure that needs to be executed only once. Subsequently, our approach involves the two standard probabilistic procedures *Gen* and *Rep*. In the following, we provide a thorough description of the three procedures, integrated by the pseudo-code of the design presented in Algorithm 2.1, 2.2, 2.3.

Init. The *Init* procedure (line 1) is responsible for generating the initial vault V . The procedure receives in input the preferred string S_{pref} , the random pool of bits B , the number of locks $|L|$ per bit-locker, and the number of bits $|C|$ to use in each XOR-ation. We provide details on the generation of S_{pref} in section 2.2.2. We first iterate over all the bits $b \in B$ (line 3), and for each b we generate $|L|$ locks (line 5). For each lock l , we select $|C|$ random indexes (line 6) and use them to select the bits in S_{pref} for the XOR-ation (line 7). The procedure eventually

returns V (line 12). The use of multiple locks adds correctness to our construction. We demonstrate this aspect in Section 2.2.4. Additionally, the XOR-ation mitigates the effects of bias and correlation in F . We provide an explanation in Section 2.2.4.

Algorithm 2.1 Init algorithm. $[]$ denotes an empty array. $X[y]$ denotes the value at index y in array X . $[x,y]$ denotes the interval of numbers from x to y .

```

1: procedure Init( $S_{pref}, B, |L|, |C|$ )
2:    $V \leftarrow []$ 
3:   for  $b \in B$ 
4:      $L \leftarrow []$ 
5:     for  $_$  in  $[1, |L|]$ 
6:        $C \leftarrow \text{drawRandomValues}([1, |S_{pref}|], |C|)$ 
7:        $l \leftarrow (\bigoplus_{c \in C} S_{pref}[c]) \oplus b$ 
8:        $L.append(l, C)$ 
9:     end for
10:     $V.append(L)$ 
11:  end for
12:  return  $V$ 
13: end procedure

```

Gen. The *Gen* procedure (line 1) generates a novel key K and its authentication token T . The procedure takes in input the reading string S_{read} , the vault V , the number of bits $|B|$ of the random pool B , and the number of bits $|K|$ of the key K . We first generate a random nonce n (line 2) and a sequence R of $|K|$ random indexes in the range $[1, |B|]$ (line 3). The indexes $r \in R$ represent the subset of bits decrypted from the vault V to form the key K . To achieve this, we perform an operation that is specular to the one executed in the *Init* procedure. For each $r \in R$ (line 5), we access all the lock values and index lists (l, C) (line 7). For each (l, C) , we XOR l with the XOR-ation of S_{read} over C and collect the resulting bit b' in X . The final bit value specific to r is then obtained as the most common value in X and stored into K' (line 11). To protect the restored values, the final value of the key K is generated via hashing K' with the nonce n (line 13). We produce a token T as the hash of key K with the list of indexes R . The generation of T adds robustness to the methodology. We discuss it in further detail in Section 2.2.4. The procedure returns the key K , the nonce n , the list of indexes R , and the token T .

Rep. The *Rep* procedure (line 1) restores the key K from the vault V . The procedure receives in input the reading string S_{read} , the vault V , the nonce n , the list of indexes R , and the token T . The procedure first iterates over $r \in R$ and restores the majority values similarly to the *Gen* procedure (lines 3 - 9). It then generates the key K'' (line 11) and the token T' (line 11). In case T' does not coincide with T , the procedure returns a null value (line 13). Otherwise, it outputs the restored key K'' as valid (line 15).

2.2.4 X-LOCK: ALGORITHM ANALYSIS

Security analysis. We first introduce two definitions that are propaedeutic to the security analysis

Algorithm 2.2 Gen algorithm. $[]$ denotes an empty array. $X[y]$ denotes the value at index y in array X . $[x,y]$ denotes the interval of numbers from x to y .

```

1: procedure Gen( $S_{read}, V, |B|, |K|$ )
2:    $n \leftarrow \text{getNonce}()$ 
3:    $R \leftarrow \text{drawRandomValues}([1, |B|], |K|)$ 
4:    $K' \leftarrow []$ 
5:   for  $r \in R$ 
6:      $X \leftarrow []$ 
7:     for  $[l, C] \in V[r]$ 
8:        $b' \leftarrow (\bigoplus_{c \in C} S_{read}[c]) \oplus l$ 
9:        $X.\text{append}(b')$ 
10:    end for
11:     $K'.\text{append}(\text{getMajorityValue}(X))$ 
12:  end for
13:   $K, T \leftarrow \text{hash}(K', n), \text{hash}(K, R)$ 
14:  return  $K, n, R, T$ 
15: end procedure

```

Algorithm 2.3 Rep algorithm. $[]$ denotes an empty array. $X[y]$ denotes the value at index y in array X . $[x,y]$ denotes the interval of numbers from x to y .

```

1: procedure Rep( $S_{read}, V, n, R, T$ )
2:    $K' \leftarrow []$ 
3:   for  $r \in R$ 
4:      $X \leftarrow []$ 
5:     for  $(l, C) \in V[r]$ 
6:        $b' \leftarrow (\bigoplus_{c \in C} S_{read}[c]) \oplus l$ 
7:        $X.\text{append}(b')$ 
8:     end for
9:      $K'.\text{append}(\text{getMajorityValue}(X))$ 
10:  end for
11:   $K'', T' \leftarrow \text{hash}(K', n), \text{hash}(K'', R)$ 
12:  if  $T \neq T'$ 
13:    return  $\perp$ 
14:  end if
15:  return  $K''$ 
16: end procedure

```

Definition 2.2.6 (Common elements between combinations). *Let X be a group of elements with cardinality $|X|$. Consider the process of drawing a random combination of y elements without replacement out of X . Then the probability p_{share} for two combinations to share at most z elements is quantifiable as $\sum_{i=0}^z \frac{\binom{y}{i} \binom{|X|-y}{y-i}}{\binom{|X|}{y}}$.*

The numerator computes the number of combinations that share exactly i elements. It selects i elements out of the y available, followed by choosing the remaining $y - i$ elements out of the remaining $|X| - y$ values. Dividing by the total number of available combinations yields the partial percentage. The summation then sums all the contributions up to z common elements.

Definition 2.2.7 (Odd binomial distribution). *Consider a Bernoulli trial with probability p of success and y independent trials. Then the odd binomial distribution considers the contributions of odd indexes in the summation of trials probabilities. In formula, $p_{\text{odd}} = \sum_{i=0}^{\frac{y}{2}} \binom{y}{2i+1} \cdot p^{2i+1} \cdot (1-p)^{y-2i-1}$*

The formula employs Bernoulli trials to assess the probability of having exactly $2i + 1$ successes in the elements. The summation from 0 to $\frac{y}{2}$ and the index $2i + 1$ permit to consider all the odd numbers between 0 and y .

We then introduce the Rouché-Capelli theorem that stands at the core of our security demonstration.

Theorem 1 (Rouché-Capelli). *A system of linear equations with n variables has a solution if and only if the rank of its coefficient matrix A is equal to the rank of its augmented matrix $A|b$. If there are solutions, they form an affine subspace $\mathbb{R}^n$ of dimension $n - \text{rank}(A)$. In particular:*

- *if $n = \text{rank}(A)$, then the solution is unique;*
- *otherwise there are infinitely many solutions.*

Proof. See introductory level book on linear algebra and geometry [139]. □

The Rouché-Capelli theorem states that a system of linear equations possesses a unique solution only if the number of independent equations equals the number of total variables. If there are more equations than variables, the system may become inconsistent, leading to the absence of a solution. Conversely, if there are fewer equations, the system becomes under-determined, providing an infinite number of potential solutions.

We are now ready to provide proof of the security of the vault V .

Theorem 2 (Solution space of the vault). *Let S_{pref} be the preferred string of F . Let V be a vault with $|L|$ locks per bit-locker and $|C|$ bits per XOR-ation. Let φ be the correlation factor related to F . Then the solution space of V has at least dimension $(1 - \varphi) \cdot |S_{\text{pref}}| + (1 - |L|) \cdot |B|$.*

Proof. The pool vault V is a collection of bit-lockers, each composed of $|L|$ locks that guard an individual bit from B (see fig 2.4). Each lock XORs a distinct bit from B with a XOR-ation of $|C|$ bits from S_{pref} . It is easy to see that a lock is a linear equation, thus making the vault V a system of $|B| \cdot |L|$ equations. Given the correlation factor φ , the independent variables provided by F to the system are $|S_{\text{pref}}| - \varphi \cdot |S_{\text{pref}}| = (1 - \varphi) \cdot |S_{\text{pref}}|$. B provides exactly $|B|$ variables as its bits are independent by definition. Thus the number of variables in the system is $(1 - \varphi) \cdot |S_{\text{pref}}| + |B|$. We then recall from theorem 1 that the dimension of the solution space is calculated as $n - \text{rank}(A)$, with n the total number of involved variables. In our case, $\text{rank}(V) \leq |B| \cdot |L|$ as some of the equations may not be independent. Hence we end up with the inequality $n - \text{rank}(V) \geq ((1 - \varphi) \cdot |S_{\text{pref}}| + |B|) - (|B| \cdot |L|)$, thus proving the statement. □

The solution space of the vault is directly related to the security of the protocol, as it determines the number of parameters a potential attacker needs to provide to solve the linear system. A closer look to theorem 2 suggests that, for increasing the security of the vault V , we want high $|S_{pref}|$, $|B|$ and low φ , $|L|$.

Definition 2.2.8 (Security of the vault). *A vault V is (α, β) -secure if the number of total locks equals α and the probability for two locks to share at most half of the elements is less than β .*

By imposing that at most half of the elements are dependent, we make sure that combining two equations results in an equation with more variables than the initial ones. We provide experimental evidence of this in section 2.2.6. The definition permits to manage the security of the vault V . Notice that the dimension of the random pool B is determined by the two parameters α, β . As per theorem 2, φ , $|S_{pref}|$, $|L|$, $|B|$ determine the solution space. φ , $|S_{pref}|$ are determined by the physical parameters of the fuzzy source F . We can then set $|L|$ and require the vault V to provide a certain probability that mixing locks does not provide information to the attacker. By doing this, we set the desired solution space and thus the dimension of $|B|$.

We can now show that the algorithm described in Section 2.2.3 satisfies the interpretation of computational fuzzy extractor given in Definition 2.2.1.

Theorem 3. *$X\text{-Lock}$ is a $(\mathcal{M}, \mathcal{F}, k, t, \varepsilon)$ -computational fuzzy extractor with error δ .*

Proof. We first notice that the helper data for a particular key in our construction is $H = (V, (n, R, T))$. The vault V is generated once and shared by all keys, while $H' = (n, R, T)$ is specific for each key. Following definition 2.2.1, we first show that (Gen, Rep) are valid functions for a computational fuzzy extractor. Procedure Gen takes in input S_{read} , V , $|B|$, $|K|$ and outputs K, n, R, T . This aligns with the definition $(K, H) \leftarrow Gen(S_{read})$, where V , $|B|$, $|K|$ are supporting input parameters and $H = H' = (n, R, T)$ for the output. Procedure Rep takes in input S_{read} , V , n, R, T and outputs either K'' or $\perp$. This again conforms to the general definition $K \leftarrow Rep(S_{read}, H)$, with $H = (V, (n, R, T))$ in the input and $K = K''$ for the output.

We now proceed in demonstrating the correctness of the algorithm. Specifically, we show that the procedure correctly retrieves the keys with the distance t between strings measured through e_{abs} and retrieval error δ proportional to $e_{bitlock}$. We focus on a single bit-locker, as the generalization is trivial. Recalling from section 2.2.3, a bit-locker is a collection of locks that XOR a bit $b \in B$ with a XOR-ation of bits from S_{pref} . By using multiple incorrect bits the probability of having an error augments. Considering the absolute error e_{abs} , we can calculate the probability e_{lock} for a lock to return the wrong value by using the odd binomial distribution in definition 2.2.7. In particular, we set the probability of success $p = e_{abs}$ and the number of trials $y = |C|$. This formulation captures the idea that a XOR-ation returns an incorrect value whenever there is an odd number of errors in its elements. Using e_{lock} , we can now estimate the probability $e_{bitlock}$ for a bit-locker to reconstruct an incorrect bit:

$$e_{bitlock} = \sum_{i=\frac{|L|}{2}}^{|L|} \binom{|L|}{i} \cdot e_{lock}^i \cdot (1 - e_{lock})^{|L|-i} \quad (2.1)$$

The final bit is calculated by majority voting. Therefore, the procedure restores the wrong bit value whenever the number of errors is greater than half of the total elements $|L|$. We use a Bernoulli trial to calculate probability of having exactly i errors, and use a summation to aggregate all the results from $\frac{|L|}{2}$ to $|L|$. To achieve the desired

level of correctness we can tune the number of locks $|L|$ to meet a given error. The same procedure is valid for all bit-lockers, thus the final error is proportional to $e_{bitlock}$, hence satisfying the initial claim.

We eventually provide proof of security of the construction. To satisfy definition 2.2.1, we need to show that the key K appears random even if a potential attacker possesses knowledge of the helper data $(V, (n, R, T))$. It is important to note that both K and T are derived using a hash function, where $K \leftarrow \text{hash}(K', n)$ and $T \leftarrow \text{hash}(K, R)$. We assume that the hash function exhibits the typical secure properties (*i.e.*, pre-image resistance, second pre-image resistance, and collision resistance). Considering all these properties together, the hash function generates outputs that appear to be random. Therefore, the pseudo-randomness of the resulting key K is conditioned solely on the amount of information leaked by the helper data $(V, (n, R, T))$. $H' = (n, R, T)$ does not leak any information on the underlying vault V : n is a random nonce, R is a random collection of indexes, and T is computed through a hash function. Regarding the vault V , we previously showed that it is (α, β) -secure under definition 2.2.8. The parameters α, β can be used to tune ϵ , thus satisfying the requirements in definition 2.2.1. $\square$

We now provide proof that X-Lock is reusable under definition 2.2.2. Reusability means that the fuzzy extractor can support multiple independent enrollments of the same value, allowing users to reuse the same source in different contexts.

Theorem 4. *X-Lock is (ρ, σ) -reusable.*

Proof. The procedure *Gen* can be run multiple times on correlated readings $S_{read}^1, \dots, S_{read}^p$ of the same source. Each time, *Gen* produces a different pair of values $(K_1, H_1), \dots, (K_p, H_p)$. The security for each extracted string K_i should hold even in the presence of all the helper strings $H_1, \dots, H_p$. In our specific case, $H = (V, (n, R, T))$. Having all H values does not compromise the security of the respective keys, as n, R are randomly determined and T is generated through hashing, making it pseudo-random by definition. Additionally, the vault V remains the same for each key, meaning that the overall reusability is only conditioned on the security of the vault V . We provide definition of the security of the pool vault in definition 2.2.8. $\square$

Notably, our model also provides insider security [130]. This means that the algorithm provides reusability even in the case where the attacker is given all the K_j for $j \neq i$. Each key K_j is the output of a hash function and provides no information about the input. Hence, a specific key cannot be used to infer information about another one, even in the presence of correlation in the input data between the two keys.

According to definition 2.2.3, a fuzzy extractor is deemed robust if a user is able to detect any tampering with the public data H . We provide proof that X-Lock is indeed a robust algorithm.

Theorem 5. *X-Lock is a τ -robust fuzzy extractor with error δ .*

Proof. The attacker may attempt to change one or more in the helper data $H = (V, (n, R, T))$. Nevertheless, any alteration would be detected by the authentication token $T \leftarrow \text{hash}(K, R) = \text{hash}(\text{hash}(K', n), R)$. Tampering with the vault V would lead to changes in the recovered values of K' , inevitably affecting the resulting hash. Furthermore, modifying the nonce n or the indexes R directly impacts the resulting hash. Additionally, changing token T to T' would require to find suitable values n', R', K'' such that $T' \leftarrow \text{hash}(\text{hash}(K'', n'), R')$. This is infeasible, given the properties of cryptographic hash functions. Consequently, the pair (Gen, Rep) strictly adheres

to the requirements specified in definition 2.2.3, as any alterations introduced by an attacker can be effectively detected through the validation token T . $\square$

Bias and correlation analysis. The scheme proposed in Algorithm 2.1, 2.2, 2.3 offers perfect secrecy when utilizing a perfectly random source F and employing uncorrelated bits to XOR with the bits from B . However, obtaining a perfectly random source using physical mediums poses considerable challenges due to issues related to bias and correlation between bits. Nevertheless, the algorithm helps in mitigating these concerns. Notice that both bias and correlation are intrinsic properties of fuzzy sources. There are works in literature proposing methodologies to reduce or even remove dependencies between bits, but they either require high computational processing or extensive analysis of the physical medium. For instance, removing correlated bits requires a perfect identification of such relationships, which is a costly and partial procedure. Missing even one correlation could be significantly harmful. Instead, our protocol implicitly mitigate bias and correlation without making assumptions over their structure. The XOR-ation adopted in the locks inherently diminishes the final bias (see Section 2.2.2). The probability p_{cbias} of an XOR-ation outputting value 1 can be represented by the equation:

$$p_{cbias} = \sum_{i=0}^{\frac{|C|}{2}} \binom{|C|}{2i+1} \cdot p_{bias}^{2i+1} (1 - p_{bias})^{|C|-2i-1} \quad (2.2)$$

The XOR-ation outputs 1 whenever there is an odd number of 1s in C . Thus, the summation between 0 and $|C|$ and the indexes $2i+1$ consider only the odd numbers. For instance, with $p_{bias} = 0.77$ and $|C| = 3$, $p_{cbias} \approx 0.579$. Consequently, an almost random source is derived from a skewed one. However, it is crucial to note that the same mechanism that reduces bias also amplifies errors. In Section 2.2.5 we provide an analysis of the cost in memory related to a particular error tolerance.

The XOR-ation also aids in mitigating the effects of correlations among bits in the source F . The correlation factor φ (see Section 2.2.2) only requires an estimate of the level of correlated bits without any specific assumption on its type. By employing it we can evaluate how the correlation impacts the overall security of the linear system. Figure 2.5 illustrates the impact of correlation on bit security with varying numbers of elements in the XOR-ation, considering $|S_{pref}| = 2^{16}$, $|B| = 2^9$, $|L| = 33$, $|C| = 4$, and $\varphi = 0.75$. We randomly drew the XOR-ation bits from the source F and measured the number of exposed bits from the random pool while bits from F were defined. The graph clearly illustrates the significant improvement achieved with an increasing number of elements in the XOR-ation. Starting with just a single element in the XOR-ation, the system fails to provide adequate security, as all the 2^9 pool bits become exposed after defining only a couple of bits. This is attributable to the relationships between locks in different bit-lockers. The setting allows one bit-locker to be entirely solved by merely defining a single bit from F , which, in turn, sets a value for all other variables in the bit locker. This creates a cascading effect that subsequently unlocks multiple other bit-lockers, thus leading to the compromised security. However, by augmenting the number of elements in the XOR-ation to three, the number of required source bits becomes closely aligned with the actual number of protected bits from the pool (as denoted by the vertical dashed line). This indicates that a potential attacker would not gain any additional advantage by exploiting the correlation between variables, as the number of variables to be defined is equivalent to the number of bits in the random pool they are attempting to set. Furthermore, when employing four elements in the XOR-ation, the system significantly surpasses the reference line, demonstrating that the impact of correlation becomes negligible in this context. The

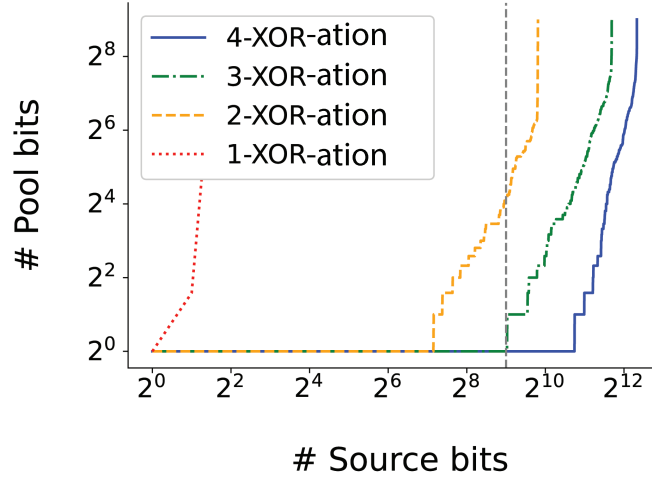


Figure 2.5: Impact of correlation on bit security with varying number of elements in the XOR cascade. The x-axis considers the number of defined bits from the source. The y-axis shows the corresponding number of exposed bits from the random pool. Using a single element provides virtually no security. From three elements onwards the number of defined bits is greater than those in the protected pool (vertical, dashed line).

higher the bias, the correlation and the security requirements to achieve adversarial non-advantageous contexts, the higher the number of bits in the XOR-ation. However, the higher $|C|$, the higher the resulting error rate in reconstructing the keys. We can estimate such value taking into account the original error rate of F .

2.2.5 COSTS ANALYSIS

Memory cost. Procedure *Init* creates the supporting pool vault. The vault V contains $|B|$ bit-lockers, one for each bit in the random pool B . Each bit-locker hosts $|L|$ locks, with each lock being a single bit. To generate the locks, the algorithm has to choose a subset of bits from S_{pref} . If chosen randomly, the subset of indexes must be stored in order to permit subsequent recovery. However, this strategy would consume a lot more memory than V itself. A more effective solution would be to design a index routine that can generate the corresponding bits on the fly. We deployed our custom strategy that works as follows. Given $n = |L| \cdot |B| \cdot |C|$ the number of indexes to be generated, compute $step = \frac{|S_{pref}|}{n}$ and identify the sequence N constituted by n indexes within the interval $[1, |S_{pref}|]$, each index $step$ away from its previous and its following. Subsequently, each bit-locker $L_i \forall 0 \leq i \leq |L|$ will be supplied with the set of indexes $\{N[i + j \cdot |L| \forall 0 \leq j \leq |C| \cdot |B|]\}$. For instance, using small numbers for sake of clarity, given $|L| = 3$, $|C| = 2$, $|B| = 3$ and $|S| = 64032$, L_0 indexes will be $\{0, 11299, 22599, 33898, 45198, 56497\}$, L_1 indexes will be $\{3766, 15066, 26365, 37665, 48964, 60264\}$ and L_2 indexes will be $\{7533, 18832, 30132, 41431, 52731, 64031\}$. A couple of remarks about this routine. First, if $step < 1$, then F cannot host n indexes, both if indexed by means of this routine and randomly chosen. Second, if $step = 1$, then N is simply a permutation of all elements of F . Last, the routine checks the presence of duplicates due to the rounding, even if with $step > 1$ duplicates should not occur. Considering the second approach, the total cost for storing the pool vault is then $|B| \cdot |L|$ bits. Procedure *Gen* generates a novel key K . It requires to store

a nonce n , an authentication token T , and the set of indexes R used to generate the key. The total cost is then $|n| + |T| + |K| \cdot \log_2(|S_{pref}|)$ bits. Procedure *Rep* does not require additional elements to be stored. The total cost is then $\mathcal{O}(|B| \cdot |L| + \mathcal{K} \cdot (|n| + |T| + |K| \cdot \log_2(|S_{pref}|)))$ bits, where $\mathcal{K}$ denotes the total number of keys generated.

Computational cost. Procedure *Init* necessitates the preferred state S_{pref} to generate the vault V . To determine the correct value, multiple readings from F are required to obtain a statistically relevant sample, often comprising hundreds of samples. The time to gather this sample may vary depending on the physical support used. For instance, **SRAMs** require a non-negligible amount of time (in the order of milliseconds) for the chip to discharge after shutdown [140]. As a result, collecting a sufficient sample might take several seconds. Once the preferred state is obtained, the procedure employs a series of XOR operations to compute the locks. Each lock utilizes $|C| - 1$ XORs for the XOR-ation and an additional XOR for the final bit. Assuming each XOR operation is $\mathcal{O}(1)$, the overall cost of computing all locks is $\mathcal{O}(|C| \cdot |B| \cdot |L|)$. Notably, procedure *Init* is only computed once, and if energy is a limited resource, it can be performed before deploying the appliance, as the vault V can be made public. Procedure *Gen* utilizes the normal state S_{read} to recover the bits from the vault and generate a new key. It then draws a random nonce p and n_K random values. Considering these operations to be $\mathcal{O}(1)$, the cost becomes $\mathcal{O}(|K| + 1)$. Subsequently, it proceeds to recover the protected bit from each lock, requiring $|C|$ XORs for each unlocking operation. This process must be performed $|K| \cdot |L|$ times, resulting in a cost of $\mathcal{O}(|K| \cdot |C| \cdot |L|)$. Finally, it computes two hashes, denoted as $\mathcal{O}(2 \cdot t_{hash})$. Thus, the overall cost is $\mathcal{O}(|K| \cdot (1 + |C| \cdot |L|) + 1 + 2 \cdot t_{hash})$. Procedure *Rep* follows a similar pattern to *Gen*. For each lock indexed by k , it unlocks the protected bit, incurring a cost of $\mathcal{O}(|K| \cdot |C| \cdot |L|)$. Additionally, it computes two hashes, yielding a total cost of $\mathcal{O}(|K| \cdot (1 + |C| \cdot |L|) + 1 + 2 \cdot t_{hash})$.

Impact of error tolerance. In Section 2.2.4, we demonstrated that by adjusting the number of locks $|L|$ within a bit-locker, it is possible to manage the error tolerance e_{lock} of the system. In this analysis, we examine the impact of e_{lock} on $|L|$ and set four different levels of tolerance as reference points. Utilizing the formula provided in Section 2.2.4, we computed the number of expected locks required to achieve a given tolerance for each specified error rate. The results are graphed in Figure 2.6.

The four lines in the graph represent distinct error tolerances. We observe that the trend is approximately linear for lower rates of errors. However, at around an error rate of 0.85, the trends start to diverge and become exponential. Notably, the curve corresponding to $e_{lock} = 1e - 05$ presents an interesting exception. Here, the number of locks exhibits an under-linear trend that extends until an error rate of 0.8. This graph effectively demonstrates that our solution experiences linear growth for low to medium error rates, resulting in linear memory and computational costs within that range.

2.2.6 IMPLEMENTATION AND COMPARISON

In order to further validate the correctness and efficiency of our approach, we implemented the algorithm on a physical test-bed using an Intel Core i5 quad-core processor with 8GB of LPDDR3 memory. We employed a **SRAM** as our **PUF**, which was easily accessible using a Keystudio MEGA 2560 R3 microcontroller board with an 8KB **SRAM**. The selection of **SRAM PUF** was motivated by several factors, including its user-friendly nature, ease of development, cost-effectiveness, and its widespread availability as a piece of hardware. To determine the preferred state of the **SRAM**, we collected 200 readings and computed the majority value for each bit. We found that 88% of the bits presented a stable value, with the remaining 12% showing an average absolute error of 0.0223 and a maximum error of 0.0455.

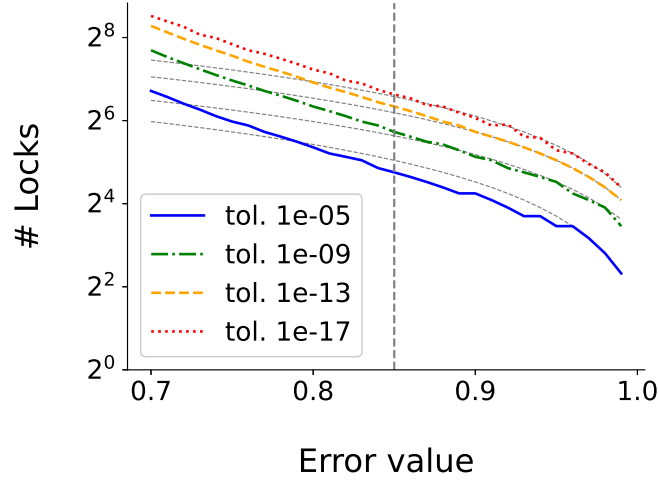


Figure 2.6: Number of locks required to achieve a given error tolerance for varying errors. The x-axis considers the error. The y-axis uses logarithmic scale and considers the number of lockers. The gray, dashed lines provides a linear reference for the four curves. The curves show an initial linear trend that slowly diverges to exponential at around value 0.85 for the error.

The algorithm was implemented in C language and compared against Woo et al. [131], which is the best-performing algorithm in terms of both memory and computational complexity. We evaluated three sets of parameters based on security levels: 80-bit, 128-bit, and 256-bit. For each level, we performed two experiments with different setups. Table 2.3 presents the parameters used for each security level.

Table 2.3: Parameters for the three security levels.

Chosen params			PUF params		Resulting params
$ K $	$ L $	$ C $	φ	e_{abs}	Exp. key error rate
80	64	2	0.75	0.15	0.0018
80	256	2	0.00	0.25	0.0025
128	64	2	0.75	0.15	0.0029
128	256	2	0.00	0.245	0.0019
256	64	3	0.75	0.10	0.0022
256	256	3	0.00	0.175	0.0013

The values for $|C|$ were carefully chosen in order to mitigate the effect of correlation as shown in Section 2.2.4. We did not consider the bias in this simulation because also the comparison work did not consider it. We decided to experiment with two specific values for the correlation factor $\varphi = 0.75$ represents a realistic correlation factor for SRAMs [138], and $\varphi = 0.00$ corresponds to a limit case where all the PUF bits are independent. In such scenario, no correlation relationship affects the bits of the PUF meaning that no XOR-ation is mandatory, net of the bias phenomenon. The correlation factor directly impacts the error tolerance by determining the number of

available equations for the linear system (see Section 2.2.4). The sixth column (*i.e.*, Exp. key error rate) reports the expected error rates in recovering the key for each configuration. We kept it at around 1 error every 500. We set $|S_{read}| = 2^{16}$ based on the size of our **SRAM** and $|B| = 2^8$ for the random pool size. Expected key error rate parameter is resulting from the choice of the other parameters. In particular, recall equation (2.1) which is the probability that a bit-locker is wrongly unlocked. Then, the probability that all bit-lockers of a given key are successfully unlocked is $(1 - e_{lock})^{|K|}$. Finally, the expected key error rate is the probability that at least one bit-locker of a given key is wrongly unlocked and it is given by $1 - (1 - e_{lock})^{|K|}$.

We measured the time needed to perform the *Rep* procedure. We performed 100000 experiments. To perform a fair comparison with the state-of-the-art, we downloaded the code of Woo et al. [131] from the public repository[†] and run it on the same appliance in order to obtain results originating from the same evaluation environment. In fact, differences affecting computational power among distinct machines could compromise the reliability of our experiments. Table 2.4 summarizes the results and compares our work with [131]. Different e_{abs}

Table 2.4: Performance comparisons

Solution	$ K $	e_{abs}	Memory	Time
Woo et al. [131]	80	0.30	$\mathcal{K} \cdot 298B$	2.3s
	128	0.20	$\mathcal{K} \cdot 543B$	7.0s
	256	0.11	$\mathcal{K} \cdot 1055B$	26.9s
Our work ($\varphi = 0.75$)	80	0.15	$2048B + \mathcal{K} \cdot 224B$	1.19ms
	128	0.15	$2048B + \mathcal{K} \cdot 320B$	1.38ms
	256	0.10	$2048B + \mathcal{K} \cdot 576B$	1.95ms
Our work ($\varphi = 0.00$)	80	0.25	$8192B + \mathcal{K} \cdot 224B$	1.12ms
	128	0.245	$8192B + \mathcal{K} \cdot 320B$	1.22ms
	256	0.175	$8192B + \mathcal{K} \cdot 576B$	1.52ms

values are arising from other parameters, such as key length $|K|$, correlation factor φ and expected key error rate. In particular, we set each e_{abs} value as the maximum we could afford in each configuration before performance degradation and security flaws significantly increased. The memory cost of the algorithms is comparable, with our implementation incurring an initial overhead for storing the vault V . However, our memory consumption grows slower with the number of generated keys $\mathcal{K}$. For instance, the memory cost for a security class of 128-bit and $\varphi = 0.75$ would become equivalent to [131] after generating 9 keys. In terms of computation time, our algorithm outperforms [131] by at least three orders of magnitude. The significant difference is due to the use of more efficient operations, mainly XORs and a limited number of hashes, in our work. On the other hand, [131] employs more resource-intensive *LWE*-based cryptography and error correction codes.

As for the error tolerance, our implementation with $\varphi = 0.75$ handles smaller amounts of error, which is a side effect of the XOR-ation mechanism (see Section 2.2.4). The correlation factor φ and the resulting number of available bits in the **SRAM PUF** significantly affect the overall performances. In fact, by setting $\varphi = 0.00$ the error tolerance of our solution becomes superior to [131] in two security levels, while being almost comparable in the third level. This setting is equivalent of having a source F with 2^{17} bits and a correlation factor $\varphi = 0.75$.

[†]https://github.com/KU-Cryptographic-Protocol-Lab/Fuzzy_Extractor

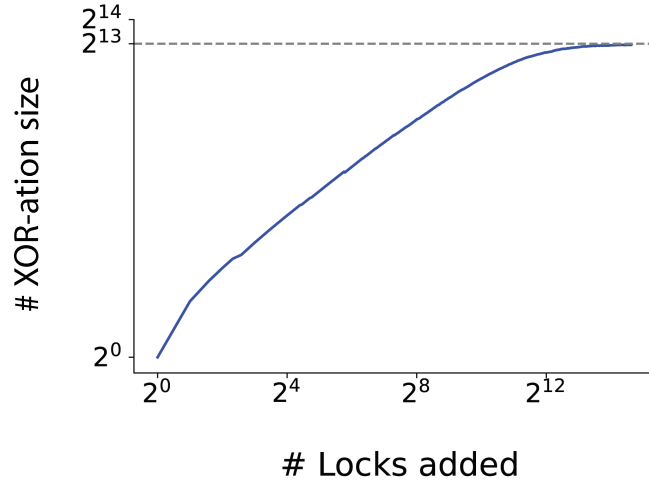


Figure 2.7: XOR-ation size while combining multiple locks. The x-axis considers the number of locks added. The y-axis considers the number of elements forming the XOR-ation. Both axes are logarithmic. The graph considers the worst case, that is the minimum number of elements in the resulting cascade.

To validate the result from Section 2.2.4, we also measured the number of errors committed in each security class. We measured 163 errors for class 80-bits, 259 for class 128-bits, and 129-bits for class 256-bits. These are in line with the expected errors calculated in Table 2.3 (respectively 180, 280, and 220) We then checked the claim made in Section 2.2.4 by assessing the capability of an adversary to combine different locks for gaining additional knowledge. We performed 100000 cycles, with every cycle randomly shuffling the locks and adding them together. We monitored the total number of elements forming the XOR-ation. Figure 2.7 plots the minimum number of elements observed in a XOR-ation by combining multiple locks. The graph clearly shows a growing trend that finds its upper bound at a value that is half of the total dimension of the source. This result validates the claim made in Section 2.2.4 on the security of the pool vault. The vast majority of locks shares no common elements, hence combining them greatly increase the XOR-ation dimension. The presence of locks with more than half common elements has little to no-impact on the overall security.

2.2.7 DISCUSSION

The proliferation of resource-constrained devices within the **IoT** domain has caused considerable challenges concerning privacy and security. Traditional methods of storing digital keys in non-volatile memory have proven intricate and costly, prompting the need for alternative solutions. This section introduces X-Lock, a novel computational fuzzy extractor specifically designed to address the limitations faced by traditional solutions in resource-constrained IoT devices. X-Lock employs a unique approach, utilizing the preferred state of a noisy source to encrypt a random string of bits, which subsequently serves as a seed to generate multiple secret keys. This design not only ensures reusability, but also effectively mitigates bias and correlation, enhancing overall robustness. To substantiate the claims of X-Lock, a comprehensive theoretical analysis was presented, encompassing security considerations and detailed implementation insights. The rigorous analysis validates the effectiveness and security of

the proposed model. To evaluate the superiority of X-Lock, extensive set of practical experiments were also conducted, and the results were compared against existing approaches. The experimental findings demonstrate the efficacy of our proposed model, showcasing its comparable memory cost (approximately 5KB for storing 9 keys of 128 bits) and remarkable speed gains, which outperform the state-of-the-art solution by three orders of magnitude (1.32ms compared to 7.0s). By offering reusability and robustness, X-Lock presents a compelling solution to the challenges posed by traditional methods of key storage. The comprehensive theoretical analysis and practical experiments affirm the superior performance of X-Lock, making it a promising advancement for enhancing privacy and security in the rapidly evolving IoT landscape.

3

Embracing complexity with multi-attribute
identities

In today’s digital landscape, the way we manage identities has evolved. Traditional methods of identity verification, such as usernames and passwords, are no longer sufficient to meet security needs. Instead, multi-attribute digital identities emerged as a crucial strategy. In an age where individuals interact with a multitude of online services, digital identities comprise a myriad of attributes and data that define them in the virtual space. The term *multi-attribute* means that digital identities are not based on a single information, but are composed of multiple attributes. These attributes may include biometric data (*e.g.*, fingerprints or facial recognition), behavioral patterns (*e.g.*, typing speed or browsing habits), and contextual information (*e.g.*, location or device used). With multi-attribute identities, verifying a user’s authenticity becomes more robust. Cyber threats, such as identity theft and fraud, can be mitigated more effectively when considering multiple authentication attributes. Multi-attribute digital identity management must also address privacy concerns. Finding the right balance between collecting sufficient data for identity verification and protecting individual privacy is a crucial challenge. For this approach to be effective, it is crucial to ensure that multi-attribute digital identities are inter-operable between different platforms and services. Compliance with data protection regulations, such as GDPR and ACFA, becomes essential. Organizations must ensure that they collect, retain and use identity attributes in accordance with legal requirements.

Section 3.1 proposes an example of multi-attribute identification on Twitter social network. Profile information such as locations, names and biographies can be used to build a machine learning model and detect the geographical location of the profile owner. The model is then applied to spot anomalies in followers’ trends during the 2020 US elections. This task can be seen as part of **RBA** [141]. **RBA** is a dynamic security approach aimed at strengthening password-centered authentication. It works by examining additional implicit factors when entering the password, such as device specificity or geolocation data. If **RBA** detects a specific risk threshold, it requires additional authentication. Endorsed by NIST’s Digital Identity Guidelines, **RBA** is implemented by numerous prominent online platforms, providing defense against security threats such as password breaches, credential-filling attacks, and weak passwords and extended guessing attempts. Section 3.2 surveys state-of-the-art solutions for the selective disclosure of attributes in multi-attribute identities. When attributes are bundled together into a single credential, the problem of disclosing only the required information arises.

3.1 LOCATION-WISE ANOMALY DETECTION ON TWITTER: UNVEILING FOLLOWERS ANOMALIES DURING THE 2020 US ELECTION CAMPAIGN

In October 2010, the University of Stanford hosted a conference titled “Liberation Technology in Authoritarian Regimes” [142]. This conference delved into the significant role that social networks played during mass protests in North Africa and the Middle East. Specifically, the discussions examined the profound impact of social media on events such as the Iranian Green Movement rallies and the mobilizations of the Arab Springs [143, 144]. It became evident that these technologies had fundamentally transformed conventional communication processes. They emerged as disruptive tools with the potential to revolutionize democratic practices at their very core. Platforms such as Facebook and Twitter have played a crucial role in facilitating a more comprehensive realization of

democratic society by providing an inclusive space for information sharing and expressing opinions.

Politics has emerged as a prominent domain that has greatly benefited from these public arenas facilitated by social media. Political and institutional entities swiftly recognized the potential of social media as a valuable tool for promoting ideas and engaging with citizens and voters. Among the various platforms, Twitter has emerged as the de facto choice for online political discourse [145]. Its user-friendly interface and expansive user base have positioned it as an ideal platform for boosting election campaigns and expanding one's electorate. Prominent political figures now utilize Twitter as a daily means to share their ideas and engage in discussions with other users. These direct interactions have played a significant role in bridging the gap between institutions and citizens, fostering greater connectivity. However, while social media facilitated more direct interactions between political figures and the public, it has also created an environment where unethical and deceptive practices can thrive.

Political accounts have been involved in manipulative practices on social media platforms, aiming to influence the discourse. The utilization of fake profiles has been observed in attempts to sway elections [146, 147, 148], impact political discussions [149, 150], suppress dissent during significant political events [150], or artificially boost the number of followers for specific accounts [151]. Recognizing the severity of this issue, numerous researchers have dedicated their efforts to proposing various solutions and detection methods [152]. However, despite the considerable amount of resources and endeavors invested, the detection of fake profiles is becoming progressively challenging due to their escalating sophistication.

This section presents an investigation into the utilization of location-wise profile categorization as a means to construct an anomaly detector for followers on Twitter. The profiles on Twitter contain valuable information such as names, locations, and descriptions, which can be leveraged to infer the provenance of profile owners. We employ this information to train a machine learning model capable of discerning profiles based on their geographical location. Subsequently, we utilize the temporal structure of Twitter elements to reconstruct the time series of followers for Joe Biden and analyze the trends across different regions. Through our time series analysis, we uncover noteworthy deviations in follower movements during the 2020 US Election campaign. These findings align with journalistic rumors [153, 154] that suggested the extensive use of fake followers within a specific time frame. Our analysis provides a broader understanding of this phenomenon by identifying potential starting and ending dates for the episode in question. Additional analyses on the extracted features corroborate our claims, showing sizable deviations for the Sub-Indian and Arabic clusters.

3.1.1 RELATED WORKS

Various websites [155, 156, 157] offer profile auditing tools on the internet. These services claim to employ extensive databases and advanced techniques, including artificial intelligence. However, there is a lack of scientific evidence supporting these claims. The algorithms used by these services are not publicly accessible, and in many cases, users are required to grant access to their personal Twitter accounts. These factors impede a reliable evaluation of the effectiveness of these tools. An intriguing alternative available online is Botometer [158], an API developed by the Observatory on Social Media at Indiana University. Botometer employs a machine learning algorithm to assess Twitter activities and assign them a score. While Botometer is widely regarded as an industry standard by experts, Yang et al. [159] uncovered significant limitations in its detection capabilities. They successfully constructed a network of social bots that evaded detection, highlighting the need for further improvements in bot detection mechanisms.

The predominant focus of academic research revolved around the utilization of machine learning methodologies for the purpose of detecting fraudulent users. The fundamental concept entails leveraging a collection of attributes (*e.g.*, , number of followers and friends, tweet frequency) to construct a classifier capable of distinguishing genuine users from fake ones. Eracsahin et al. [160] used a supervised discretization technique named Entropy Minimization Discretization (EMD) to classify the data points and evaluated the result using the Naïve Bayes algorithm. Cresci et al. [1] analyzed diverse online resources and compiled a comprehensive set of common detection criteria. They proceeded to assess these criteria using a labeled dataset, ultimately developing a robust random forest model by incorporating easily accessible features. Analogously, Chu et al. [161] constructed a random forest classifier using a manually labeled ground truth. The use of random forests is advantageous due to their interpretability. Nonetheless, Castellini et. al [162] demonstrated that the accuracy of random forests may deteriorate over time. They replicated the experimental setup established by Cresci et al. [1] and applied it to a more recent dataset, observing a significant decline in performance. To address this issue, they proposed an approach based on denoising autoencoders. This model learns to reconstruct the features of authentic accounts, thereby enabling the identification of anomalous users through examination of the reconstruction error. Recently, Khalil et al. [163] conducted a survey encompassing various machine learning techniques, which they subsequently applied to a labeled dataset. The results varied across different approaches, highlighting the importance of selecting suitable methodologies. Although machine learning-based methodologies have demonstrated favorable outcomes, their practical applicability is restricted. Castellini et. al [162] revealed that the performance of these models deteriorates rapidly over time due to the evolving characteristics of fake accounts. Furthermore, constructing a sizable labeled dataset is necessary for training these models. The quality and comprehensiveness of these datasets significantly impact the categorization results.

A different line of research focused instead on clustering and graph centrality measures. Stringhini et al. [164] applied clustering techniques on the user relationships in a dataset composed by premium fake users and randomly sampled ones. Mehrotra [165] instead used graph centrality measures and artificial neural networks random forest to classify the users. These methodologies possess the advantage of being platform-agnostic, meaning they are not reliant on specific parameters associated with a particular platform. However, collecting the relationship graph typically demands vast amounts of resources in terms of data collection and time investment.

3.1.2 BACKGROUND

Twitter [166] is a popular micro-blogging service that was launched in July 2006. It has established itself as one of the major social media platforms, boasting a reported active user base of approximately 230 million users [167]. The platform’s core feature is the tweet, which allows users to share their thoughts and opinions in short text messages of up to 280 characters. In addition to posting tweets, users have the option to provide additional information on their profiles. This includes a short biography, a profile image, location, and birthday. The platform’s friendship functionality is unidirectional, meaning that users can freely follow other users without requiring a reciprocal connection. Consequently, Twitter differentiates between friends and followers. A user’s friends are the accounts they follow, while their followers are the accounts that follow them. As a result, each user maintains separate lists of both friends and followers within their profile.

In June 2010 Twitter engineers made an announcement [168] regarding the open-sourcing of their internal ID generation system called Snowflake [169, 170]. A Snowflake ID is a 64-bit number composed of three parts. The

leading 42 bits represent a timestamp indicating the creation time, followed by 10 bits for the machine code that generated the ID, and the remaining 12 bits for a rolling sequence number. Every object within Twitter, including entries in the friends and followers lists, is associated with a unique Snowflake ID. This organization allows for the chronological ordering of the lists and facilitates efficient pagination. The lists are divided into batches of IDs, with each batch containing a collection of user IDs along with references to the previous and subsequent batches. These references are referred to as the previous cursor and the next cursor. The list is consumed in a reverse chronological order, starting from the newest followers and progressing towards the oldest ones. Consequently, the next cursor points to an older batch of IDs. To collect the required data, we utilized the official Twitter API service [171]. This allowed us to efficiently collect all the necessary data for our study. Python [172] was selected as the programming language for both the data collection and statistical analysis due to its user-friendly nature and extensive range of supporting libraries. Python provided a convenient and flexible environment for handling Twitter data and conducting subsequent analysis.

The data collected with the API enables us to extract features and run statistical analyses on them. In this section, we focus on the Class A features proposed by Cresci et al. [1]. To simplify the process of detecting fake followers, they propose a set of features that can be easily collected with the Twitter API. They then use an information fusion-based sensitivity analysis technique to assign a normalized score that estimates the importance of a single feature. We summarized the features and their scores in table 3.1. We highlighted in bold the features that we will use in our analyses. We discarded the features *following rate* and *duplicate profile pictures* because impossible to calculate with the data at our disposal. We also removed the features *friends/followers $\simeq 100$* and *number of followers* because of their redundancy. We eventually eliminated the feature *bot in biography* as our goal is to detect illicit bots that try to evade detection. For the sake of simplicity, we assigned an acronym to the chosen features that will be used throughout the section.

3.1.3 BUILDING A LOCATION-WISE CLASSIFIER

Twitter profiles offer a wealth of information that can be utilized to assess the geographical location of the profile owner. Users typically provide real names as their profile names, which can serve as a valuable indicator. Additionally, users often include descriptions in their profiles that offer insights into their lives, including general references that may aid in the geolocation process. Furthermore, Twitter profiles may also include a location field, although this information is less commonly provided, appearing in approximately one-third of the profiles. However, this location data has proven to be highly advantageous in categorizing profiles based on their region.

Labeled dataset and character-level embedding. We initially curated a dataset comprising profiles that were labeled based on their country of origin. To automate this process, we implemented a strategy of searching for country-specific profiles on Twitter and collecting information about their followers. By focusing on profiles that are associated with local realities and post in the local language, we made the assumption that a significant portion of their followers would also consist of individuals from those particular countries. Table 3.2 provides a summary of the countries included in our study, along with the corresponding profile and the total count of followers collected. The selection of countries was based on their significant presence and activity on the Twitter platform [173]. We chose to steer clear of excessively large accounts to minimize the potential for an abundance of fake profiles. The underlying principle is that smaller accounts are generally more likely to have an authentic follower base. We excluded the United States and the United Kingdom from our analysis as our focus was on

Rank	Feature	Score	Acronym
1	friends/(followers²) ratio	1.000	ffs
2	age	0.919	age
3	number of tweets	0.816	twt
4	profile has name	0.782	nam
5	number of friends	0.781	frs
6	has URL in profile	0.768	url
7	following rate	0.765	
8	default image after 2 months	0.755	img
9	belongs to a list	0.752	lst
10	profile has image	0.751	prf
11	friends/followers ≥ 50	0.736	f50
12	bot in biography	0.734	
13	duplicate profile pictures	0.731	
14	2 $\times$ followers $\geq$ friends	0.721	2ff
15	friends/followers $\simeq 100$	0.707	
16	has address	0.677	adr
17	no bio, no location, friends ≥ 100	0.664	bld
18	has biography	0.602	bio
19	number of followers	0.594	

Table 3.1: Class A features as proposed in Cresci et al. [1]. In bold the features we considered in our analysis. The column acronym summarizes the shortcut we use throughout the section to reference the specific feature.

exploring less commonly examined regions.

In our preprocessing stage, we focused on converting the textual information from the dataset (*i.e.*, names, descriptions and locations) into numerical representations suitable for inputting into a machine learning model. To achieve this, we developed a character-level embedding model. Using characters as the fundamental units for the embedding was a deliberate choice, considering that the texts we dealt with were relatively short (typically one to three words long). This approach has been successfully applied in various text analysis tasks by previous studies [174, 175] which have shown comparable results to more conventional word-level methods. For training the character-level embedding model, we utilized the widely-used Python library called *Gensim* [176]. *Gensim* is renowned for its capabilities in topic modeling, document indexing, and similarity retrieval. To construct the embedding model, we utilized the textual information available from the followers of the country-specific profiles. We applied a filtering process to exclude characters that fell outside the Unicode range of $0x0000-0xffff$. This allowed us to focus on characters from major alphabets used worldwide. To address the issue of rare characters, we substituted infrequent letters with the symbol $\sim$. This step was crucial since rare characters, which occurs less than five times overall, have limited impact and can potentially yield poorer results. We eventually trained a *Word2Vec* model using the cleaned and pre-processed data. *Word2Vec* class is accessible from *gensim.models*. We opted for an embedding representation size of 32 components. This means that each character would be represented as a

Country	Profile	Followers
Argentina	MindefArg	223337
Brasil	ItamaratyGovBr	361624
Canada	CitImmCanFR	177253
Colombia	CMILANOTICIA	504396
Egypt	Extranewstv	504396
France	AssembléeNat	458957
Germany	AuswaertigesAmt	504392
India	pibhindi	409959
Indonesia	kemenhub151	473630
Japan	MofaJapan_jp	478826
Malaysia	501Awani	504395
Mexico	PartidoMorenaMx	395766
Pakistan	Naya__Pakistan_	504388
Philippines	TV5manila	461416
Saudi Arabia	KSAMOFa	504386
South Korea	nylee21	233691
Spain	MAECgob	294312
Thailand	TNAMCOT	493076
Turkey	GelecekPartiTR	259409

Table 3.2: Twitter profiles used for creating the location-wise dataset. The countries were selected based on the associated user base on Twitter. Each profile was chosen for its specificity to the respective country. We focused on profiles with moderate-sized follower bases to reduce the risk of encountering numerous fake accounts.

vector of 32 components, capturing its semantic properties and relationships with other characters. We employed a window size of one character, enabling the model to consider the immediate context by associating each character with the characters that directly followed it in the text.

The resulting character-level embedding comprised a total of 11890 vectors, each representing a specific character in the dataset. One notable characteristic of the embedding is that characters belonging to the same alphabet exhibit closer vector representations compared to characters from different alphabets. This phenomenon reflects the similarity and shared linguistic characteristics within each alphabet. Moreover, the embedding was able to capture other linguistic structures beyond just alphabet divisions. Figure 3.1 shows the vector representation for the Latin alphabet projected to a 2D-space using t-distributed Stochastic Neighbor Embedding (t-SNE algorithm). Within the Latin alphabet, the embedding revealed that vowels had closer representations to one another compared to consonants. This encoding of proximity among vowels captured the distinct role that these letters assume in the language.

Training the convolutional model. Our research study embraced the utilization of a character-level convolutional network as the architectural framework for our model. The primary driver behind this choice was the limited length of the available training texts. Since profiles’ names, descriptions, and locations are typically con-

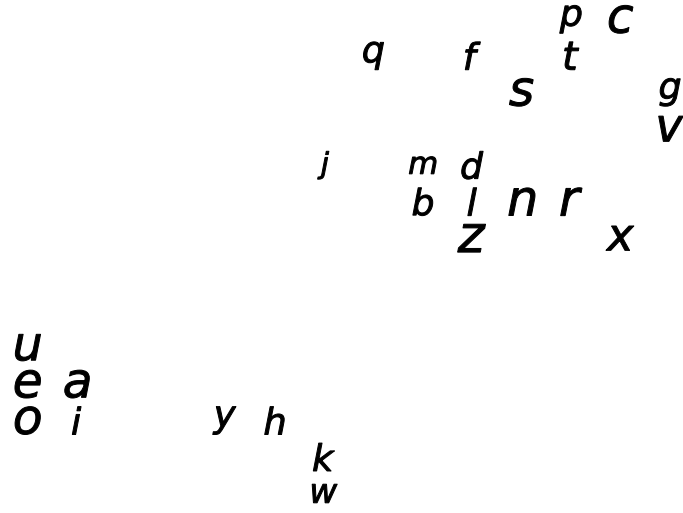


Figure 3.1: Embedding for the Latin alphabet projected to 2D space using t-SNE. The vowels are grouped together, showing that the model captured more complex structures.

cise, character-level models offer a well-suited approach for capturing crucial patterns and features at a detailed level. Our choice aligns with previous research conducted by Zhang et al. [175], who extensively explored the use of character-level convolutional networks for text classification tasks. Their empirical investigation demonstrated the effectiveness of character-level models in achieving state-of-the-art performances.

In our training process, we generated three separate models, each focusing on a specific aspect of the Twitter profiles: names, descriptions, and locations. To prepare the input data for these models, we needed to convert the variable-length strings into fixed-length inputs suitable for the convolutional network. To determine the appropriate fixed lengths for each input type, we considered various factors. For names, we set the fixed length to 50 characters, which aligns with the maximum length allowed for names on Twitter. For descriptions, we set the fixed length to 96 characters. This choice was informed by a statistical analysis of the data set, which indicated that over 95% of descriptions fell within this length limit. For descriptions longer than 96 characters, we truncated them to fit within this constraint. Similarly, for locations, we set the fixed length to 64 characters. Again, this decision was based on the statistical analysis, which revealed that the majority of locations were well below this length limit. In all cases, strings shorter than their limit were padded with the $\sim$ symbol that we previously used to represent infrequent characters.

The structure of our training model was designed as follows:

- **Input.** The model takes as input a matrix with a shape of $(\text{limit}, 32)$, where the specific value of `limit` depends on the type of string being processed (50 for names, 64 for locations, and 96 for descriptions). Each character in the input string is encoded using the character-level embedding, resulting in a 32-dimensional vector representation.
- **Convolutional Layers.** The model comprises three convolutional layers. These layers utilize kernel sizes of 9, 7, and 5 respectively. All convolutional layers employ 64 filters, *ReLU* activation, *valid* padding, and

a random normal kernel initializer with a mean of 0.0 and standard deviation of 0.05;

- **Output.** The output of the convolutional network is flattened and connected to two fully connected layers. The first fully connected layer uses *ReLU* activation, and the second layer uses *softmax* activation to ensure the output neurons activations sum to one. A dropout layer with a dropout rate of 0.5 is applied between these final layers for regularization. The dimensions of these two layers depend on the number of outputs required for the specific classification task at hand.

Initially, we trained a model focused on predicting individual countries. Each country listed in table 3.2 was considered separately, necessitating the utilization of 19 output neurons. Although the overall results of this model were satisfactory, we observed that it encountered difficulties in accurately distinguishing between countries that share similar backgrounds. Notably, the model struggled with correctly classifying Spanish-related countries (*i.e.*, Mexico, Colombia, Argentina, and Spain), as well as South-East Asian countries (*i.e.*, Malaysia, Indonesia, and Thailand). To address these challenges, we made the decision to aggregate countries and develop a second model. The selection of country clusters was guided by the insights gained from the performance of the first model. Remarkably, the composition of the clusters were also supported by cultural similarities among the countries, further reinforces the validity of our modeling approach. As a result, the second model was designed with 11 output neurons, corresponding to the number of country clusters identified. Table 3.3 summarizes the clustered entities. Both models were trained using *categorical cross-entropy* as loss function and *validation loss* as control metric. We used *Adam* with a learning rate of 0.001 as the optimizer.

Cluster	Countries
Arabic	Egypt, Saudi Arabia
Filipino	Philippines
French	Canada, France
German	Germany
Sub-Indian	India, Pakistan
South-East Asian	Indonesia, Malaysia, Thailand
Japanese	Japan
Korean	South Korea
Portuguese	Brasil
Spanish	Argentina, Colombia, Mexico, Spain
Turkish	Turkey

Table 3.3: Clusters of countries used to train the second model. The clustering was suggested by the results of the first model and are further supported by cultural similarities.

Table 3.4 provides a comprehensive overview of the key metrics obtained from evaluating the three trained models. To assess the performance of our models for the multi-class classification task, we calculated separate accuracy, precision, sensitivity, and specificity measures for each cluster. Overall, the scores obtained from the evaluation are generally favorable. However, there are notable exceptions worth mentioning. Specifically, the model trained on descriptions exhibited the poorest performance. Both the Filipino and Portuguese clusters dis-

played low precision and accuracy. These inaccuracies can be attributed to cultural similarities between these clusters and other groups. The Filipino cluster was frequently misclassified as part of the Southeast Asian cluster, while the Portuguese cluster was predominantly misclassified as part of the Spanish cluster. A similar pattern is observed in the model trained on names, albeit with better overall results, particularly for the Portuguese cluster. It is worth noting that the best-performing model was the one trained on locations, which demonstrated optimal results across all clusters.

In light of the favorable scores obtained from the individual models, we proceeded to combine them into a straightforward evaluation pipeline. The first model used in this pipeline was the location model, which served as our primary reference. For profiles that lacked a specified location or received a low categorization score from the location model (*i.e.*, < 0.33), we turned to the name model for further evaluation. Similarly, in cases where the name model did not provide a label or yielded a low score (*i.e.*, (*i.e.*, < 0.50)), we resorted to the description model (with threshold ≥ 0.33). Profiles that did not receive any label throughout this pipeline were marked as unlabeled. The selection of thresholds involved employing the Kneedle algorithm [177] to pinpoint the inflection points in the distributions generated through model application to the data.

Cluster	Accuracy	Precision	Sensitivity	Specificity
Arabic	0.98	0.94	0.89	0.99
Filipino	0.98	0.0	0.0	1.0
French	0.97	0.85	0.8	0.99
German	0.98	0.91	0.63	1.0
Sub-Indian	0.96	0.77	0.88	0.97
South-East Asian	0.95	0.8	0.92	0.96
Japanese	0.99	0.97	0.96	1.0
Korean	1.0	0.98	0.92	1.0
Portuguese	0.93	0.0	0.0	1.0
Spanish	0.91	0.74	0.97	0.89
Turkish	0.99	0.91	0.9	1.0

Table 3.4: Confusion matrix for the description model.

3.1.4 REVERSE ENGINEERING TWITTER TIMESTAMPED LISTS

In section 3.1.2 we introduced Snowflake’s IDs as timestamp-based identifiers used by Twitter. Each object within the platform, including entries in the followers list, is associated with one of these IDs, which likely encodes the object’s creation time. Notably, the IDs in the followers list appear in descending order, implying that the list is organized chronologically from the most recent follower to the older ones. In this context, the next cursor can be seen as a time pointer that coincides with the ID of the last entry in the current batch.

Preliminary analysis. To qualitatively validate this assumption, we performed experiments by incrementing and decrementing the next cursor value to observe the changes in the returned batches. Figure 3.2 illustrates the observed dynamics. By decrementing the next cursor (*i.e.*, moving to an older time), we consistently obtained the

Cluster	Accuracy	Precision	Sensitivity	Specificity
Arabic	0.98	0.9	0.89	0.99
Filipino	0.99	0.98	0.88	1.0
French	0.97	0.87	0.84	0.99
German	0.99	0.94	0.86	1.0
Sub-Indian	0.97	0.88	0.95	0.98
South-East Asian	0.99	0.97	0.96	0.99
Japanese	0.99	0.96	0.94	1.0
Korean	1.0	0.95	0.89	1.0
Portuguese	0.99	0.96	0.84	1.0
Spanish	0.97	0.9	0.97	0.97
Turkish	1.0	0.93	0.92	1.0

Table 3.5: Confusion matrix for the location model.

same batch of new IDs. Conversely, by incrementing the cursor (*i.e.*, moving to a more recent time), we again obtained the same batch of IDs, but with the ID of the last entry in the current batch appearing at the top. This behavior strongly suggests that the followers’ list is organized as a time series, where the cursors serve as convenient time pointers. When requesting a batch of IDs, the system utilizes the cursor as a starting point and navigates backward through the time series until enough IDs are retrieved or the list ends. Consequently, two consecutive cursors can be used to temporally frame a batch of IDs. The time resolution of this frame depends on the time distance between the two cursors. Throughout our data collection process, we maintained a resolution under one hour.

Regressing a conversion formula. In section 3.1.4 we demonstrated that the next cursor in Snowflake IDs corresponds to the ID associated with the last user in the returned batch. By leveraging this information, we can establish a conversion formula between cursors and UTC timestamps by creating a mapping between these two values. This mapping can be constructed by repeatedly requesting batches of length one from the top of a followers’ list and recording the corresponding timestamps and next cursor values whenever a new follower is encountered. To estimate the actual UTC timestamp, we measured the time elapsed between two consecutive API calls during this data collection process. The average delta between these measurements was found to be 1.18 seconds, with a standard deviation of 0.38 seconds. To balance the uncertainty introduced by this time measurement, we opted to use the average of the two values as an estimation of the real UTC timestamp. To expedite the collection of data points, we monitored an influential Twitter account, specifically JoeBiden, as it gains hundreds of new followers within a short period. We collected a total of 696 data points between February 13th and February 16th, 2023. As mentioned in Section 3.1.2, we focused on the 42 most significant bits of the cursors for the regression calculation. To determine the best-fit line for the cursor-to-UTC timestamp conversion, we utilized the RANSAC regressor. We found the formula for the fitted line to be:

$$utc = 0.004 * (cursor \gg 22) - 1826 \quad (3.1)$$

The right bit-shift operation (denoted by $\gg$) extracts the first 42 bits of the cursor value. The regression

Cluster	Accuracy	Precision	Sensitivity	Specificity
Arabic	0.97	0.92	0.92	0.98
Filipino	0.97	0.97	0.05	1.0
French	0.97	0.82	0.62	0.99
German	0.98	0.91	0.56	1.0
Sub-Indian	0.97	0.86	0.94	0.97
South-East Asian	0.97	0.88	0.91	0.98
Japanese	0.99	0.96	0.95	1.0
Korean	1.0	0.96	0.94	1.0
Portuguese	0.97	0.95	0.26	1.0
Spanish	0.93	0.76	0.98	0.91
Turkish	0.99	0.87	0.96	0.99

Table 3.6: Confusion matrix for the name model.

analysis of the cursor-to-UTC timestamp mapping yielded an extremely high R^2 of approximately 1, indicating that almost all of the variance in the dependent variable is explained by the independent variable. The RMSE of the regression was 0.38, further demonstrating the accuracy and reliability of the fit. Notably, the regression resulted in a negative intercept, suggesting that the reference epoch is half an hour before the Linux epoch (1970-01-01 00:00:00). However, given the substantial time span covered by the formula and the small error introduced by our procedure, we speculate that the actual starting epoch coincides with the Linux epoch. Therefore, we can omit the intercept term from the conversion formula. The formula tells us that increasing by 1 the leading 42 bits equals an increase of 0.004s in time. In other words, the formula has a sensitivity of 4ms.

3.1.5 RESULTS ANALYSIS

In section 3.1.3 we presented three convolutional neural networks that were designed to infer the location of Twitter profiles based on some text features. In section 3.1.4 we derived a formula that enables us to reconstruct the timeline of followers for a given Twitter account. We combined these two tools to conduct an analysis of the region distribution in the followers' timeline of Joe Biden, the current President of the United States, whose Twitter handle is @JoeBiden. Our analysis specifically focused on the year 2020, which was marked by the presidential elections. We collected the complete followers timeline in the week between February 23rd and February 28th, 2023. Each profile in the timeline was categorized based on the classification rules outlined in section 3.1.3. We analyzed the data and plotted the trend over time for different regions, considering the proportion of each cluster in relation to the total number of profiles.

Figure 3.3 illustrates the resulting graph, with a focus on the four major regions that emerged from our categorization process (*i.e.*, Arabic, French, Sub-Indian, Spanish). To enhance understanding, we marked two significant dates: the day Joe Biden announced Kamala Harris as his vice president (August 11th, 2020) and the general election day (November 3rd, 2020). Upon examination, we observe that both the Spanish and French groups exhibit a relatively stable trend throughout the entire year, with minor fluctuations occurring in the month preceding

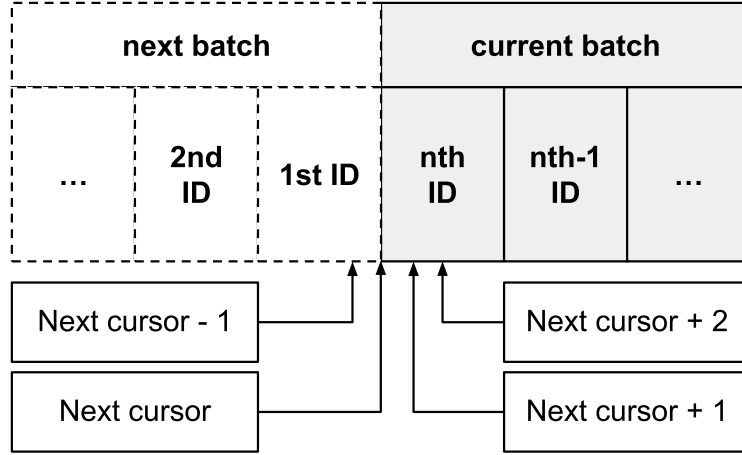


Figure 3.2: Cursors act as time pointers in the followers timeline. When a cursor retrieves a batch of IDs, the system returns a list starting from the ID that temporally follows the cursor. Decrementing the cursor does not change the result, while incrementing it returns a list with the last ID of the current batch on top.

and following the election day. In contrast, the Arabic and Sub-Indian groups display distinct patterns. Both groups experience a notable surge in activity concentrated between July and August. Following this period, the Arabic group exhibits a gradual and consistent increase in followers during the last three months of the year. On the other hand, the Sub-Indian group undergoes a second substantial surge that commences two weeks prior to the general election, persists beyond it, and maintains a significant share of followers until the end of the timeline.

The analysis of Joe Biden’s followers timeline in 2020, particularly regarding the Sub-Indian group, aligns with journalistic reports suggesting the presence of Indian fake followers during that period [153, 154]. These reports claim that the unusual provenance of the followers may indicate their questionable origin. The articles includes interviews with individuals allegedly involved in supplying fake followers on commission. Interestingly, our graph also demonstrates a significant increase in Sub-Indian profiles during August 2020, which corresponds to the time frame mentioned in the article. It is worth noting that the rise in followers starts a month prior to the announcement of Kamala Harris as the vice presidential candidate. This finding challenges the explanation that the surge in Indian profiles is solely attributed to the announcement and Kamala Harris’s Indian origins. The peak finds its origin on June 12nd, 2020. We investigated and found an article [178] published in that day from the New York Times indicating Kamala Harris as the emerging candidate for the vice-presidency of Joe Biden. This may have been the trigger that started the upward trend for the Sub-Indian cluster. However, we notice a similar pattern discernible from the Arabic group. While a mere trend analysis cannot definitively determine the malicious source behind this surge, it does serve as compelling evidence warranting further investigation.

A similar occurrence takes place in close proximity to the election day. Both the Arabic and Sub-Indian groups return to their pre-burst levels during September. Subsequently, the Arabic group gradually increases in size without any evident bursts, potentially indicating the presence of legitimate followers. On the other hand, the Sub-Indian group experiences a second sudden surge that initiates two weeks prior to the election day. This surge persists until the latter half of November, stabilizing at a level twice as high as the original count until mid-December, after which it gradually decreases. This particular spike in the Sub-Indian group can partially be explained by the

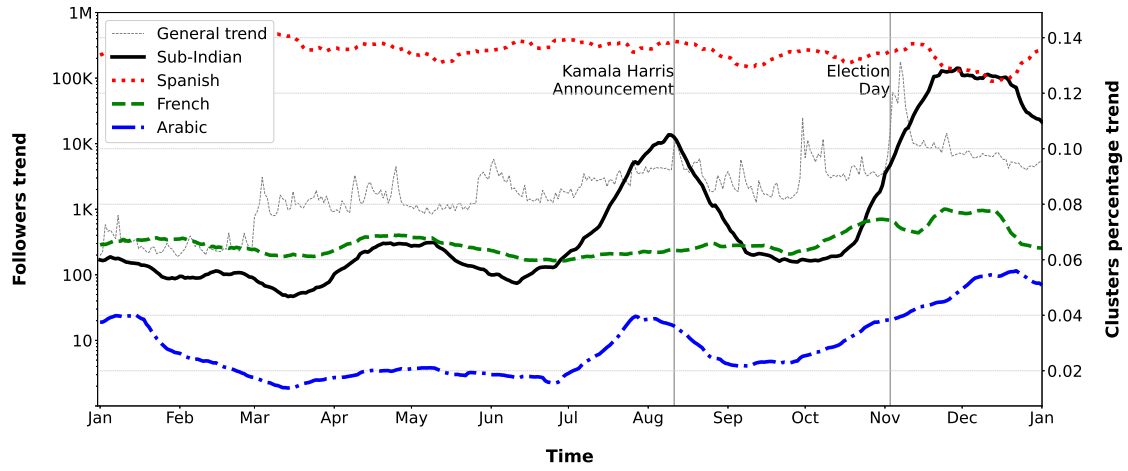


Figure 3.3: Followers trend of @JoeBiden divided by ethnicity during 2020. The thinner, gray line represents the followers trend in logarithmic scale. The solid black line illustrates the percentage trend of Sub-Indian profiles. The dotted red line represents the percentage trend of Spanish profiles. The dashed green line portrays the percentage trend of French profiles. Lastly, the dash-dotted blue line displays the percentage trend of Arabic profiles. The anomalous trends detectable before the announcement of Kamala Harris as vice-president supports the rumors of an usage of Sub-Indian fake followers and suggest an additional usage of Arabic fake followers. Another anomaly emerges for the Sub-Indian group approximately two weeks before the Election day, further supporting the presence of fraudulent activities.

resonance of the election event. The overall trend clearly exhibits a surge of followers in the days following the event, driven by the resonance of the occasion that attracted a significant number of users. However, it is noteworthy that the spike in Sub-Indian profiles starts two weeks prior to the election day, during a period when the general trend did not display any distinctive movement. This observation suggests that the surge in Sub-Indian profiles is somewhat anomalous and may potentially indicate fraudulent behavior.

To attempt to unravel the complexity and provide evidence of the anomalous nature of the peaks, we conducted additional analyses based on the class A features as outlined in table 3.1. Our focus was directed toward the period from July 21, 2020, to August 21, 2020, representing the peak month during the summer of the announcement. We categorized the features into continuous and binary, employing two distinct analytical methods. Figure 3.4 illustrates the graphs for the four continuous features, (*i.e.*, ffs, age, twt, and frs). We derived the Kernel Density Estimation (KDE) from region-segmented data, utilizing Gaussian Kernels as the algorithm and Scott's Rule for bandwidth estimation. To enhance clarity, we applied logarithmic scaling to the x-axis in some instances, simplifying data interpretation.

What immediately stands out is the divergent behavior of the Arabic cluster. The trend for ffs leans towards positive values, indicating a propensity to have a significantly higher number of followings compared to followers. The account age at the time of follow (*i.e.*, age) is concentrated at values less than an hour. The number of tweets tends towards either null or relatively small values, typically within the range of 10 to 100. All of these characteristics serve as indicators of anomalous accounts, suggesting the presence of potentially fake accounts created specifically to inflate followers' lists. A similar line of reasoning can be applied to the Sub-Indian cluster. In fact, it exhibits a similar behavior to the Arabic cluster concerning ffs and frs. Regarding age and twt, while the magni-

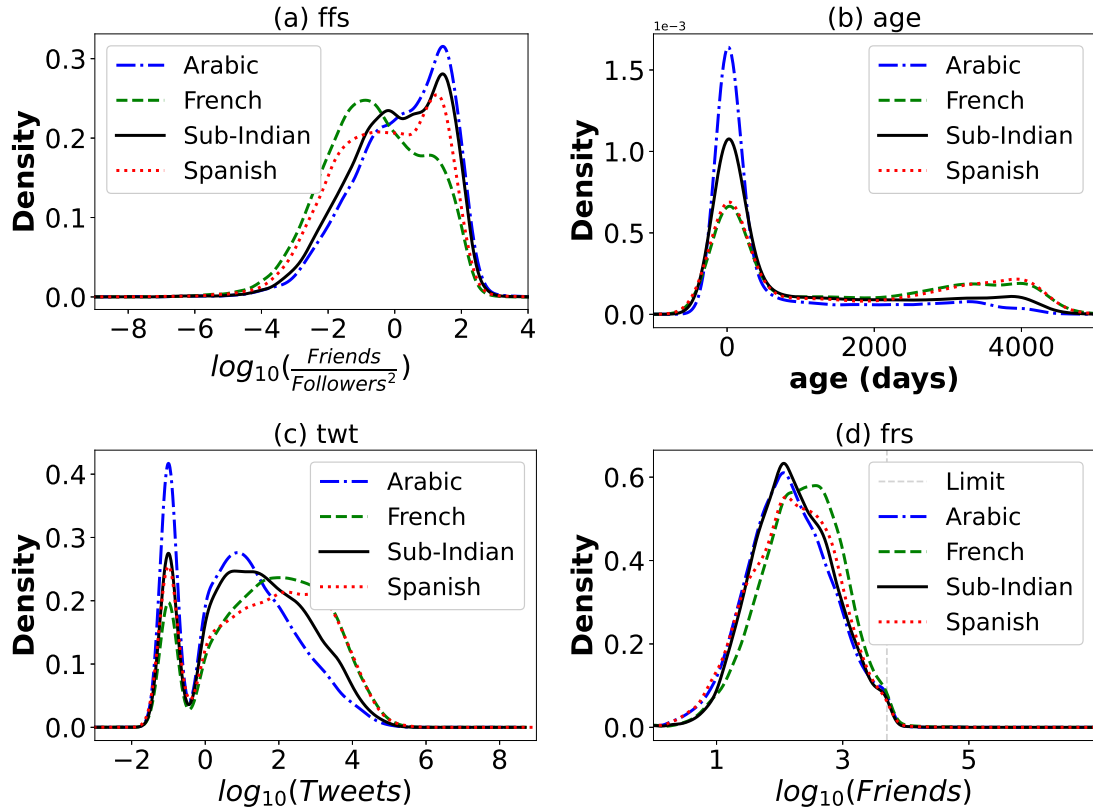


Figure 3.4: Kernel density regions of 4 class A continuous features from [1]. The data refers to the period between July 21st, 2020 and August 21st, 2020. Graph (a) depicts the logarithm of the ratio between the number of friends and the number of followers squared. Graph (b) shows the age of the accounts. Graph (c) displays the logarithm of the number of tweets. We represent 0 values with -1 . Graph (d) depicts the logarithm of the number of friends. The 4 lines show the trend of the locations of reference.

tude of the curves may differ somewhat, the general trend mirrors that of the Arabic cluster, leading to analogous conclusions.

Conversely, the French cluster exhibits distinctive characteristics. *ffs* displays a specular trend compared to the Arabic cluster, favoring negative values. The age trend is significantly lower, with younger ages comprising only half of the Arabic cluster's values, and it features a long tail, indicating the presence of older accounts. Additionally, the number of tweets favors higher values, typically ranging from 100 to 1000, with a larger number of friends *frs* at an average of 1000. These characteristics align with the behaviors of regular social media users. An intriguing case arises with the Spanish group, which appears to manifest a combination of features from both profiles. While *ffs* closely resembles the Arabic trend, age more closely aligns with the French trend. Further analyses are required to comprehend the nature of such behaviour.

Table 3.7 presents statistics regarding the binary variables. It displays the fraction of the population within a cluster for which a specific feature is True (False). Firstly, it is noticeable that *nam*, *url*, and *bdl* did not yield informative results. This implies that the imposed division led to either a definite classification (e.g., True 1.0,

Table 3.7: Class A boolean features from [1]. The numbers represent the fraction of cluster population that satisfies (does not satisfy) the relative feature. The data refers to the period between July 21st, 2020 and August 21st, 2020.

Feature	Arabic	French	Sub-Indian	Spanish
nam	Not informative			
url	Not informative			
img	0.17(0.83)	0.09(0.91)	0.12(0.88)	0.14(0.86)
lst	0.08(0.92)	0.30(0.7)	0.15(0.85)	0.26(0.74)
prf	0.95(0.05)	0.80(0.2)	0.90(0.10)	0.79(0.21)
f5o	0.19(0.81)	0.09(0.91)	0.17(0.83)	0.10(0.9)
2ff	0.06(0.94)	0.15(0.85)	0.08(0.92)	0.14(0.86)
adr	0.34(0.66)	0.66(0.34)	0.50(0.50)	0.48(0.52)
bdl	Not informative			
bio	0.62(0.38)	0.74(0.26)	0.71(0.29)	0.64(0.36)

False 0.0) or a completely random classification (i.e., True 0.50, False 0.50). The table corroborates the previously observed trends. The Arabic cluster exhibits anomalies that support the hypothesis of low-quality profiles. Arabic profiles display a higher prevalence of default profiles (img, prf), inferior following-followers ratios (f5o, 2ff), lower list memberships (lst), and fewer instances of addresses and biographies (adr, bio). In contrast, the French cluster exhibits more favorable characteristics indicative of healthy profiles. The Sub-Indian cluster still tends towards the Arabic group, with low values for (lst, bio) and high values for (prf, f5o). The Spanish group continues to strike a balance between the Arabic and French clusters.

In summary, the additional analysis of class A features has led to the recognition of the Arabic cluster as a potential source of fake profiles. The Indian cluster also exhibits characteristics that raise doubts about the quality of its profiles, although the divergence is less pronounced than that of the Arabic group. Conversely, the French cluster has provided a baseline reference due to its positive qualities. The Spanish cluster remains a subject of uncertainty, warranting further investigation to ascertain its nature more definitively.

3.1.6 DISCUSSION

Social media has transformed traditional communication and hold the potential to revolutionize democratic practices. Political entities recognized the value of social media for promoting ideas and engaging with citizens, with Twitter becoming a prominent platform for political discourse. However, social media also became a breeding ground for unethical practices, including fake accounts and deceptive practices. This section examined the viability of employing ethnicity-based analysis to detect anomalies among the followers of political figures. To achieve this, we developed a machine learning algorithm capable of categorizing Twitter profiles according to their ethnicity. Subsequently, we reconstructed the timeline of followers for Joe Biden and conducted an analysis of the trends observed across different ethnic groups. The findings of our study provide compelling evidence supporting the presence of fake followers and align with the reports from journalistic sources, thereby corroborating their claims. Despite significant efforts from researchers to combat fraudulent activities, the issue of political manipu-

lation remains highly pertinent. Social networks have achieved widespread global penetration, and ensuring their impartiality and objectivity is of paramount importance. In this context, our intention is to make a meaningful contribution toward these objectives. We aspire to contribute to the ongoing efforts aimed at safeguarding the integrity and fairness of social networks, ultimately fostering a more trustworthy and unbiased digital environment.

3.2 CRYPTOGRAPHIC METHODS FOR SELECTIVE DISCLOSURE IN VERIFIABLE CREDENTIALS

In today’s interconnected digital landscape, the concept of identity has undergone a transformative evolution. Traditional centralized identity systems often present significant issues regarding privacy, security, and control over personal information. However, emerging technologies have paved the way for a paradigm shift toward **SSI**, **VCs**, and **SD**, giving individuals unprecedented authority and agency over their digital identities. This section explores the multiform world of **SSI**, exploring the revolutionary concepts of verifiable credentials and selective disclosure. It seeks to clarify the fundamental principles, technology frameworks, and real applications that underpin these concepts, aiming to provide a comprehensive understanding of their implications for privacy, security, and user autonomy.

The advent of blockchain technology, **DIDs** and **VCs** has allowed people to exercise more control over their digital identities. This shift towards **SSI** introduces a break from conventional identity systems, allowing individuals to manage, control and selectively share their credentials and personal data, thus redefining the dynamics of trust and identity verification in the digital world. Through the lens of **SSI**, we aim at exploring the mechanisms that allow people to assert their identities without depending on intermediaries, improving privacy and promoting a safer, user-centered digital ecosystem. It also investigates the concept of **VCs** such as digitally signed attestations, revolutionizing the way credentials are issued, shared, and verified in a decentralized manner. In addition, the concept of **SD** emerges as a crucial aspect of **SSI**. It allows people to exercise granular control over the information they disclose, ensuring that only the necessary and relevant attributions are shared, thereby reducing the privacy risks associated with full disclosure of data.

3.2.1 SELF-SOVEREIGN IDENTITY

The National Institute of Standards and Technology (NIST) defines digital identity as “the unique representation of a subject engaged in an online transaction. A digital identity is always unique in the context of a digital service but does not necessarily need to uniquely identify the subject in all contexts” [179]. As noted by Cameron [180], the Internet was originally built without an identity layer. It was developed as a military project to interconnect machines and share resources across multiple networks. The interconnection was made using only the address of the machines without any reference to the identities of those managing the machinery. Initially, that was not a problem given the size of the original Internet. There were only a restricted number of academic scientists accessing it, many of whom knew each other. But now, with billions of users and multiple billions of devices wired in, the issue has become of capital importance.

According to a study carried out by the Pew Research Center [181], 91% of Americans agree that people have lost control over their personal information online. 80% shared their concerns about businesses accessing social media data, and 64% agreed on a stricter regulation from the govern. These statistics are justified by the abundant number of cyber incidents that resulted in massive exposure of personal data. Only the last months of 2022 have experienced some of the largest data breaches in history. In September, Optus revealed that about 10 million customers had their personal data stolen in Australia’s worst data breach [182]. In November, a hacker posted on a dark web forum a data dump allegedly containing 500 million WhatsApp numbers [183]. In December, a hacker asked for a ransom to delete Twitter’s private details of more than 400 millions users [184]. These recent examples highlighted the main problem of the currently deployed identity models: they do not place users in immediate control of their identity data, resulting in limited control or visibility. Indeed, the majority of online service providers support their own specific set of data management policies and practices, resulting, in some cases, in the impossibility for users to access their own data at any time, for reasons they deem justified

On February 2012, Devon Loffreto wrote about a “Sovereign Source Authority” in one of his online articles [185]. In it, the author postulates the established right for each individual to own an identity, and expresses the idea that national registration nullifies that ownership. This is one of the first references of self-sovereign identity (SSI), a novel model of identity that recently emerged onto the Internet.

The need for sovereignty The right to personal identity is internationally recognised through a range of declarations and conventions [186, 187]. Holding a personal identity is seen as a crucial step for every human being towards her development and realization as an independent entity. Consequently, personal identities should be universally valid, non-revocable and under the strict control of its owner. However, current identification systems do not meet the above expectations.

Traditional, state-issued credentials like passports or driver licenses usually have nation-wide validity and can be revoked anytime. This problem acquired its own dignity during the 2015’s European migration crisis. Over 5 million refugees came to the continent to request asylum [188], and the absence of a common identification framework resulted in many people lacking a proper identity.

On the same page, the digital world suffers from centralized identity control and balkanization of domains. Each website tends to build its own identification system that barely interacts with the others. Even worse, the users are at the complete mercy of site owners who can revoke the credentials at their will without further notice. Steps toward a greater decentralization of control (*e.g.*, federated identity [189], user-centric models [190].) did not solve the centralization and revocability issues.

SSI systems want to put a solution to these problems. The digitalization of the world is offering a great opportunity to change the old paradigms and make route to better identification models. These systems emphasize the centrality of the user in the administration of her own identity. This is not only limited to the interoperability of the identity across multiple locations, but it also extends in creating full autonomy from external entities.

Founding principles Several sources [191, 180] provided a list of principles that SSI systems should provide. These principles were further elaborated by the Sovrin Foundation [192] and organized into three main categories.

1. **Security.** Identities should be *persistent* and long-lived. The adopted system must enforce the *protection* of personal data, and minimize its disclosure to the essential.
2. **Controllability.** The owner of an identity must have complete *control* over it. The identity should *exists* independently from other entities, and its usage must always require the user’s explicit *consent*.

3. **Portability.** Identities must be *portable*. There should be *interoperability* between different providers, and the overall system must be *transparent*. Each user must be able to *access* her stored data.

The points delineate an identity management system where individuals gain full control over their digital identity. At the same time, they recognize that an identity can be used for both beneficial and harmful purposes. SSI should balance between transparency, fairness, and community support with the protection of the individual.

3.2.2 VERIFIABLE CREDENTIALS

A **VC** is a machine-readable type of credential that complies with the specifications set out in [193]. It has cryptographic security and respects privacy concerns. **VCs** are linked to a **DID** within a **DID** Document, establishing a connection with an individual's identity. See Figure 3.5 for a **VC** sample. **VCs** are generated by the broadcaster and transmitted to the recipient. These credentials consist of a series of statements relating to various attributes, such as name, date of birth, academic grade, identification number or other details that the issuer intends to associate with the recipient. To share a specific statement with a Verifier, a presentation is created. Presentations allow the selective disclosure of attributes, for example by revealing the date of birth while hiding the individual's name.

Identity as a set of attributes. The core of **SSI** implementations are the **VCs**. **VCs** are an inter-operable data structure suitable for representing cryptographically verifiable claims. They can be seen as the digital version of conventional credentials such as passports, licenses, and credit cards. A credential typically consists of a collection of information such as its type, subject, issuing authority, specific attributes, and constraints. **VCs** reflect this setting and port it to the digital realm. Each **VC** contains a set of attributes which have been signed so that the owner can exhibit them as proofs of a claim. It also includes links to the subject of the claim and to the entity that issued it. The digital format enables the usage of cryptographic signatures that render the credential verifiable (*i.e.*, tamper-evident and trustworthy). Additionally, it permits the usage of privacy-enhancing technologies (*e.g.*, **ZKP**) to protect the data from unauthorized access.

A generalized model. The **W3C** formally issued a recommendation for **VCs** [193]. Figure 3.5 depicts the proposed architecture. The *holder* is the owner of one or more **VCs**. These credentials are created and issued by an *issuer* to certify some claims about the *holder*. The *holder* stores the **VCs** inside a personal *wallet*, and exhibit them to a *verifier* when requested. The issuing and verification of **VCs** operates through an external *verifiable data registry* which stores relevant data to accomplish the task (*e.g.*, identifiers, public cryptographic keys, **VC** schemas, revocation registries). The recent advent of the blockchain technology provided a potential solution to distribute the *verifiable data registry* and not returning to a centralized model [194, 195, 196].

DIDs [197, 198]) are a useful integration to promote decentralization and distribution. **DIDs** are a new type of identifier that have been designed to be decoupled from centralized registries. They function as unique identifiers linking to a digital structure containing related information, the **DID** document. In the context of **VCs**, a **DID** may be used to reference the entities involved. Each **DID** would resolve in a **DID** document containing cryptographic material and other related metadata. *Issuers* would include their **DIDs** and the *holders* ones into the **VCs**. *Verifiers* could verify the correctness of the claims by accessing the **DID** documents appended to the decentralized *verifiable data registry*.

Privacy concerns. The digitalization of credentials poses inevitable questions about privacy and data protection. The convenience of machine-readability renders **VCs** more susceptible of being collected and cross-analyzed to in-

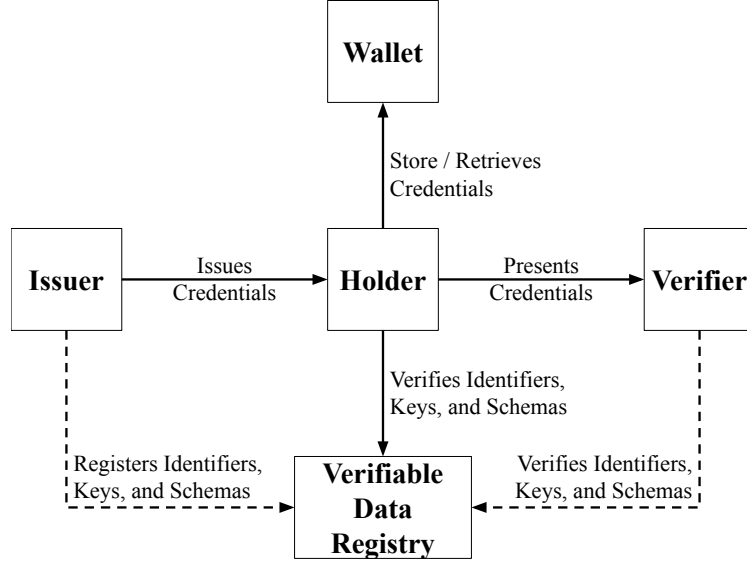


Figure 3.5: General architecture for a Verifiable Credentials system.

for additional information. Depending on the available credentials, a third party may perform explicit and implicit correlation tasks.

The **W₃C** model provides the field *credentialSubject* for those fields susceptible to privacy violations. Government-issued identifiers, shipping addresses, full names are some examples of information that can easily determine a *holder*. Other explicit correlation data may be the signature used to anchor the claims to their *holder*. Device fingerprinting may provide correlation hints external to the VCs themselves. Information like IPs, web-browser metadata, mobile GPS position can provide enough evidences.

One way to cope with the problem is to use abstract claims whenever possible. For example, requesting that the age is over a certain threshold is a stronger protection with respect to asking the age of the birth date of the *holder*.

In general, the accepted best practice for preventing data correlation is the principle of data minimization. Already required by multiple jurisdictions (HIPAA [199], GDPR [200]), it means that the exposed data should be minimal and used specifically for the intended purpose. The *issuers* should limit the content of a VC to the minimum required by the claim. The *verifiers* should narrow the scope of requested claims to the minimum necessary for accessing the service. One way to enforce data minimization by design is the usage of selective disclosure algorithms for VCs (see section 3.2.5). With selective disclosure, the *holder* is no longer forced to expose all the attributes of a VC and can make fine-grained decisions about what information to share.

3.2.3 CRYPTOGRAPHIC PRELIMINARIES

We denote $\{0, 1\}^l$ the set of binary strings of length l , that is the set of integers in the interval $[0, 2^l - 1]$. Similarly, $\pm\{0, 1\}^l$ denotes the integer interval $[-2^l + 1, 2^l - 1]$. We denote the set of non-negative integers as $\mathbb{Z}^*$. Cryptography makes an extensive use of prime numbers. A prime number is a integer number $p > 1$ that is divisible only

by 1 and itself. We usually denote prime numbers with the letters p, q . The first prime numbers are 2, 3, 5, 7, 11, and it is possible to demonstrate that the list continues indefinitely. Prime numbers are important because they constitute the atomic factors of all the integer numbers. The process of finding the prime factors of a number is called factorization, and the difficulty of efficiently performing it lays the foundation of some modern cryptographic schemes (see section 3.2.3). A particular type of primes that will find application in later sections are the safe primes.

Definition 3.2.1 (Safe primes). *A prime number p is safe if $p = 2p' + 1$, where p' is prime.*

Modular arithmetic for a modulus n defines any two elements a, b that differ by a multiple of n to be equal. In other terms, an integer a is equivalent to $b \in [0, n - 1]$ if there is a value k such that $a - b = kn$. We denote this operation as $a \equiv b \pmod{n}$, and b is often called the residue of the operation. For example, $38 \equiv 14 \pmod{12}$ because $38 - 14 = 2 \times 12$. It is noteworthy that for any prime number p , there always exists a multiplicative group of integers modulo p . A particular type of residues that will find application in some cryptographic methods are the quadratic residues.

Definition 3.2.2 (Quadratic residues). *$QR_n \subseteq \mathbb{Z}_n^*$ denotes the set of quadratic residues modulo n , that is $a, b \in \mathbb{Z}_n^* | b^2 \equiv a \pmod{n}$.*

In this work, we will sometimes use the idea of computational intractability. An intractable problem is a problem in which the only exact solution is one that takes too many resources (e.g., time, memory). In other words, a problem in which no efficient solution exists. Often, this solution is a brute-force-styled solution. Cryptographic protocols make extensive use of problems that are considered intractable (e.g., factorization in section 3.2.3, discrete logarithm in section 3.2.9).

Group theory. The concept of group is central to abstract algebra and lies at the very core of current cryptographic methods.

Definition 3.2.3 (Group). *A group is a non-empty set G associated with a binary operation $\cdot$ that combines any two elements $a, b \in G$ and returns another element $c \in G$, i.e., $a \cdot b = c \in G$. The operation $\cdot$ must satisfy three group axioms:*

- **Associativity.** $\forall a, b, c \in G (a \cdot b) \cdot c = a \cdot (b \cdot c)$
- **Identity element.** $\exists e \in G : \forall a \in G e \cdot a = a \wedge a \cdot e = a$. Such e is unique.
- **Inverse element.** $\forall a \in G \exists b \in G : a \cdot b = e \wedge b \cdot a = e$, where e is the identity element. Such b is unique. We denote it as a^{-1} .

A particular example of group that finds applications in cryptography is the cyclic group. For any element $g \in G$, we define a cyclic subgroup generated by g as the subgroup $\langle g \rangle = g^k | k \in \mathbb{Z}$ consisting in all the integer powers of g . In this context, we also define the order $|g|$ of g as the number of elements found in $\langle g \rangle$. We can now define cyclic groups.

Definition 3.2.4 (Cyclic group). *A cyclic group is a group which is equal to one of its cyclic subgroups. In notation, $\exists g \in G : \langle g \rangle = G$. We call element g the generator of G .*

A notable example of cyclic group under the operation of multiplication is the set of positive integers modulo n that are relatively prime to n . In particular, the group is always cyclic with order $\phi(n)$ when the modulo is a prime number p . We may denote this group as $\mathbb{Z}_p^*$.

Rivest-Shamir-Adleman assumptions. In 1978 Rivest et al. [201] proposed the first description of a public-key cryptosystem. Commonly known as **RSA** algorithm, the security of its construction relies on the difficulty of factoring the product of two large prime numbers $n = pq$.

Definition 3.2.5 (RSA modulus). *Let n be a $2k$ -bits number and p, q be k -bits prime numbers. Then n is a **RSA** modulus if $n = pq$.*

The core of the algorithm rests on the practicality of calculating three large integers (e, d, n) such that $(m^d)^e \equiv m \pmod{n}$. e becomes part of the public key (e, n) used for encrypting a message, while number d is kept private for decryption. The easiest way to decrypt a message without knowing the secret d is to factorize the **RSA** modulus n , but no efficient algorithm to perform it has been found to date.

Definition 3.2.6 (RSA assumption). *Given an **RSA** public key (n, e) and a random integer $u \in \mathbb{Z}_n^*$, it is hard to compute $v : v^e \equiv u \pmod{n}$.*

A stronger version of the **RSA** assumption states that the **RSA** setup remains hard to break even in the case the attacker can freely choose the public number e .

Definition 3.2.7 (Strong RSA assumption). *Given an **RSA** modulus n and a random integer $u \in \mathbb{Z}_n^*$, it is hard to compute to compute $e > 1, v$ such that $v^e \equiv u \pmod{n}$.*

The cryptographic methods surveyed in this work make use of a particular type of **RSA** modulus termed special.

Definition 3.2.8 (Special RSA modulus). *A **RSA** modulus $n = pq$ is special if p, q are safe primes.*

Consider p, q safe primes and g a generator of quadratic residues modulo $n = pq$. In this case, it can be shown that knowing the order of g is equivalent to determining the factorization of n .

Diffie-Hellman assumptions. The Diffie-Hellman assumptions formally define the hardness of solving certain problems involving the calculation of discrete logarithms in cyclic groups. For given real numbers a, b , the logarithm $c = \log_b a$ is a number such that identity $b^c = a$ holds. Analogously, in cyclic group G we can define powers $b^k \forall k \in \mathbb{Z}$, and the discrete logarithm $c = \log_b a$ is the integer value k such that $b^k = a$.

Definition 3.2.9 (Discrete logarithm). *Let G be a cyclic group under the multiplication operation, and let a, b be any element of G . For any positive integer k , b^k denotes the product of b with itself k times. Similarly, let b^{-k} denote the product of b^{-1} with itself k times. The discrete logarithm is then the value k that satisfies the identity $b^k = a$.*

A general algorithm for computing the discrete logarithm is to brute-force the value of k by exponentiating the base b to increasingly larger values. However, the running time of the solution is linear in the order of the group, thus exponential in the number of bits. There exist more sophisticated algorithms (e.g., Baby-step, Giant-step [202]), but none of them runs in polynomial time. These considerations permit to formally define hard problems based on the calculation of discrete logarithms.

Definition 3.2.10 (Decisional Diffie-Hellman assumption). *Consider a multiplicative cyclic group G with generator g of order $|g|$. Given g^a, g^b with $(a, b) \in \mathbf{Z}_{|g|}$, the value g^{ab} cannot be distinguished from a random element in G . More formally, the following two probability distributions are computationally indistinguishable:*

- (g^a, g^b, g^{ab}) with $(a, b) \in \mathbf{Z}_{|g|}$ randomly chosen;
- (g^a, g^b, g^c) with $(a, b, c) \in \mathbf{Z}_{|g|}$ randomly chosen.

The **DDH** assumption [203] is related to the discrete logarithm as an efficient algorithm for calculating it would permit to calculate $\hat{a} = \log_g(g^a)$ and then check $z \stackrel{?}{=} (g^b)^{\hat{a}}$ with z coming from tuple (g^a, g^b, z) .

Definition 3.2.11 (Computational Diffie-Hellman assumption). *Consider a multiplicative cyclic group G with generator g of order $|g|$. Given $a, b \in [0, |g| - 1]$ and tuple (g, g^a, g^b) , then calculating the value g^{ab} is computationally intractable.*

The **CDH** assumption relates with the discrete logarithm problem because an efficient algorithm to would permit to calculate $\hat{a} = \log_g(g^a)$ and then to retrieve $g^{ab} = (g^b)^{\hat{a}}$ starting from the tuple (g, g^a, g^b) . The **CDH** assumption is a weaker assumption as computing g^{ab} from (g, g^a, g^b) directly translates to solving the **DDH** assumption.

A variation of the aforementioned assumptions is the **q-sDH** assumption [204].

Definition 3.2.12 (q-sDH assumption). *Consider G_1, G_2 being cyclic groups of prime order p , let g_1, g_2 be generators of G_1, G_2 and $x \in \mathbf{Z}_p$. Given the tuple $(g_1, g_2, g_2^x, \dots, g_2^{x^q}) \in G_1 \times G_2^{q+1}$, it is infeasible to compute the pair $(c, g_1^{\frac{1}{x+c}}) \in G_1 \times \mathbf{Z}_p$.*

Bilinear maps and pairings. A bilinear mapping refers to a function that operates on elements from two distinct vector spaces, producing an element belonging to a third vector space. This function exhibits linearity in each of its input arguments. Bilinear mappings serve as a fundamental tool within the domain of pairing-based cryptography, facilitating the establishment of relationships between cryptographic groups. In this context, we are mainly interested into cyclic groups of prime order.

Definition 3.2.13 (Bilinear map). *Let G_1, G_2 be groups of prime order p . A bilinear map $e : G_1 \times G_2 \rightarrow G_t$ is a function that $\forall u \in G_1, \forall v \in G_2, (a, b) \in \mathbf{Z}$ guarantees the identity $e(u^a, v^b) = e(u, v)^{ab}$.*

For practical purpose, we require e to be non-degenerate (i.e., $\exists(u \in G_1, Q \in G_2) : e(u, v) \neq 1$, with 1 the identity value of G_t). Additionally, we require e to be admissible, meaning that e is efficiently computable. G_1, G_2, G_t are isomorphic groups since they are cyclic and have the same order. They are different groups in the sense that we represent the elements and compute the operations differently. However, it is usually the case that $G_1 = G_2 = G$. In modern cryptographic applications, G is typically an elliptic curve. These groups are based on the Weil and Tate pairings over elliptic curves [205].

The core properties of pairings can be expressed in the equations: $e(g^a, g^b) = e(g^b, g^a) = e(g^{ab}, g) = e(g, g^{ab}) = e(g, g^a)^b = e(g, g)^{ab}$. Cryptographic pairing leverages elliptic curves to achieve these properties, therefore the notation g^n represents a generator point on a curve added to itself n times instead of a multiplicative group generator. The survey in [206] provides a starting point for exploration of cryptographic pairings.

3.2.4 PROTOCOLS FOR ZERO-KNOWLEDGE PROOF OF KNOWLEDGE

ZKPs offer a multitude of applications, seamlessly weaving together various concepts, including: 1. *private data validation*, such as confirming that a person has enough money in their bank account or comparing DNA profiles without revealing the complete genetic sequences; 2. *anonymous authorization*, such as validates that an applicant has the right to access a restricted area of a website without revealing his identity or demonstratig membership of a list of approved countries or states without specifying the exact one; 3. *outsourcing calculations*, such as delegating an intensive calculation of resources or switching from a blockchain model where everyone performs the same calculation to one where a single part performs the calculation and all the others verify. Although these applications may seem tempting at first glance, the underlying methodology is a mathematical marvel in the fields of mathematics and cryptography. It has been the subject of research for over three decades since its inception in 1985 [207], and subsequently marked by the introduction of non-interactive proofs [208]. In any **ZKP** system, two key actors emerge: the prover and the verifier. This framework is developed in three distinct phases. During the witness phase, the prover calculates a piece of evidence that encapsulates their claim and then transmits it to the verifier. In the challenge phase, the verifier asks the demonstrator several questions regarding the proof. Eventually, in the answer step the prover provides answers to these questions, which the verifier uses to accept or reject the proof generated. At all these stages, three crucial properties hold:

- **Completeness.** The verifier will invariably be persuaded by the prover's truthful statement. If the prover can demonstrate to the verifier that their claim is valid, the verifier shall invariably accept the evidence generated.
- **Soundness.** This property indicates that the prover cannot convince the verifier that their false claim is, in fact, true, except with a negligible probability. If the prover's claim is misleading, the verifier shall reject the generated evidence.
- **Zero-knowledge.** This attribute ensures that the prover does not transmit any useful information to the verifier. Basically, if the demonstrator can prove to the verifier the validity of their claim, the verifier only acquires knowledge that the demonstrator's statement is true.

This section introduces various protocols to prove knowledge over quantities and relations. To describe these protocols, we use the notation proposed by Camenisch and Stadler [209]. As an illustrative example, we describe the syntax with a **ZKP** of knowledge of a discrete logarithm, also known as Schnorr's identification protocol [210].

$$PK\{(\alpha) : y = g^\alpha\} \quad (3.2)$$

The Greek letters denote the values of which the prover is demonstrating the knowledge, while the other parameters are publicly known. The right side summarizes the relations that the secrets must satisfy (*i.e.*, $y = g^\alpha$). The implementation details are shown in the following diagram. The value $X \equiv g^x \pmod q$ is called *protocol commitment*. After performing the algorithm, the prover has successfully demonstrated the knowledge of x to the verifier, while not disclosing its value.

A potential generalization of protocol 3.8 is to employ a set $g_1, \dots, g_l$ of generators and demonstrate the knowl-

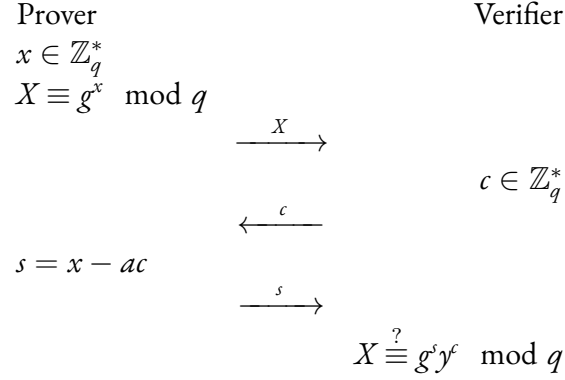


Table 3.8: Schnorr's Identification scheme. g, q, y are the public parameters. x is the prover's secret

edge of a representation of a discrete logarithm (see section 3.2.3).

$$PK\{(\alpha_1, \dots, \alpha_l) : y = \prod_{i=1}^l g_i^{\alpha_i}\} \quad (3.3)$$

The generalized protocol works in the same fashion as its simpler version. The prover's secrets are now $x_1, \dots, x_l$, and each of them is associated with a specific generator $g_1, \dots, g_l$. The commitment then becomes $X = \prod_{i=1}^l g_i^{x_i}$ and the protocol remains unchanged. The syntax also provides the symbol $\wedge$ to combine different proof protocols.

$$PK\{(\alpha_1, \dots, \alpha_l, \beta_1, \dots, \beta_{l'}) : y = \prod_{i=1}^l g_i^{\alpha_i} \wedge z = \prod_{i=1}^{l'} h_i^{\beta_i}\} \quad (3.4)$$

The corresponding protocol is obtained by running the two proofs of knowledge in parallel as sub-protocols. The prover produces all the commitments for both the proofs and forward them to the verifier. The verifier then replies with a single challenge message, that is the verifier chooses the same challenge for both protocols. The final verification checks that the equations of both the sub-protocols hold. This last protocol can be easily extended for an arbitrary number of secrets.

By combining the protocols introduced so far, we can now derive a series of useful proofs of knowledge.

Equality of discrete logarithms. The proof convinces a verifier that $\log_b z$ equals the first element in the representation of y (i.e., α).

$$PK\{(\alpha, \beta) : y = g^\alpha b^\beta \wedge z = b^\alpha\} \quad (3.5)$$

Prover calculates $t_y = g^{r_\alpha} b^{r_\beta}$, $t_z = b^{r_\alpha}$. Verifier then checks that $t_y = y^c g^{s_\alpha} b^{s_\beta}$, $t_z = z^c b^{s_\alpha}$, with $s_\alpha = r_\alpha + cx_\alpha$, $s_\beta = r_\beta + cx_\beta$.

Product of discrete logarithms. The proof convinces a verifier that $\log_g y_3$ is the product of $\log_g y_1$ and $\log_g y_2$.

$$PK\{(\alpha, \beta) : y_1 = g^\alpha \wedge y_2 = g^\beta \wedge y_3 = y_1^\beta\} \quad (3.6)$$

Power of discrete logarithms. The proof convinces a verifier that $\log_g y_2$ is equal to $(\log_g y_1)^2$.

$$PK\{(\alpha) : y_1 = g^\alpha \wedge y_2 = y_1^\alpha\} \quad (3.7)$$

Non-zero discrete logarithm. The proof convinces a verifier that α is non-zero when $\log_g h$ is unknown to the prover.

$$PK\{(\alpha, \beta) : g = y^\alpha h^\beta\} \quad (3.8)$$

Up to now, the protocols used groups of prime order q that is known to both the involved parties. There are however particular cases in which the security of the protocol depends directly on the knowledge of q . This is the case of RSA groups, a subset of $\mathbb{Z}_n^*$ where n is the product of two safe primes (see section 3.2.3). Let us consider protocol 3.8 modulo a composite.

$$PK\{(\alpha) : y = g^\alpha \pmod n\} \quad (3.9)$$

In this case, the public information are y, g, n, l_n , with l_n the bit-length of n , and the prover's secret is $x = \log_g y$. The difference with the original protocol is that now the prover does not know the factorization of n . To securely compute the initial commitment, the prover considers $x \in \{0, 1\}^{l_n + l_c + l_\emptyset}$, where l_c is the bit-length of the verifier's challenge and $l_\emptyset$ is a security parameter ($e.g., l_\emptyset = 80$).

A secret lies in a given interval. One useful property of a composite modulo is that the prover cannot reduce the response message modulo the order of the group. Thus the verifier can argue about the bit-length of the secret known to the prover.

$$PK\{(\alpha) : \frac{y}{g^{(a+b)/2}} = g^\alpha \pmod n \wedge \alpha \in \{0, 1\}^{l_n + l_c + l_\emptyset}\} \quad (3.10)$$

The prover knows a value $x \in [a, b]$ such that $y = g^x \pmod n$. The protocol follows the same procedure as equation 3.9, with only the verifier adding a final check that $s \in \pm\{0, 1\}^{l_n + l_c + l_\emptyset}$. It is straightforward to generalize the protocols just discussed in the same way as the protocols we considered for groups of known order q .

We close this section by introducing the Fiat-Shamir heuristic [211]. In the original definition of **ZKP** [207], the proof was interactive. This means that the prover and the verifier engage in a two-way communication, to establish the validity of the proof. Interactive **ZKP** can be complex and require multiple rounds of communication, which can be limiting in some applications. The Fiat-Shamir heuristic is a method of converting interactive **ZKP** into non-interactive ones. The idea is to replace the verifier's challenge in the interactive protocol with a hash of the verifier's messages. This eliminates the need for the verifier to actively participate, making it non-interactive. The safety of non-interactive **ZKP** is based on the underlying collision resistance of the hash function used. As a practical example, consider construction 3.8. To make it non-interactive, the verifier is replaced by computing the challenge as $c = \mathbb{H}(g||y||t||m)$, with $m \in \{0, 1\}^*$ and $\mathbb{H}$ being an hash function. The signature of m then becomes the pair (s, c) . A similar transformation can be applied to all the other **ZKP** schemes.

3.2.5 SELECTIVE DISCLOSURE SCHEMES

It is considered best practice for issuers to atomize information in a credential or use a signature scheme that allows for selective disclosure [193]. However, one of the key capabilities **ZKP** (section 3.2.4) are required to provide is the selective disclosure of claims in a **VC** without requiring the issuance of multiple atomic **VCs**. Selective disclosure

indicates the ability of a holder to make fine-grained decisions about what information to share. In the remainder of this section, we will introduce three major selective disclosure schemes that has been proposed in recent years.

Camenisch-Lysyanskaya scheme. The **CL** signature scheme is named after its creators Jan Camenisch and Anna Lysyanskaya, whose work [209, 212, 213, 214] has expanded through a series of articles published between 2001 and 2004. The **CL** signature has gained notoriety in the context of Anoncreds [215], and has also played a key role as a cryptographic base in the Idemix [216] and Hyperledger Indy [217] projects. Originally, the **CL** signature was built on the robust **RSA** assumption (see section 3.2.3). As a result, the size of its keys was very long to ensure adequate security against attackers. To address the issue, significant improvements have been made through the application of pairing-based encryption (see section 3.2.3). For the sake of clarity, this section will focus on the classic construction based on the **RSA** assumption. The core element in the **CL** scheme is the signature protocol [213, 218]. It allows to apply an efficient discrete logarithm proof protocol to prove the knowledge of a signature and to retrieve the signatures on secret messages. The structure of the scheme is shown in protocol 3.9. In this protocol, the resulting signature is the tuple (e, A, v) and it permits the prover to authenticate a group of messages $M = \{(m_0, \dots, m_{L-1}) : m_i \in \pm\{0, 1\}^{l_m}\}$. Notably, the authors in [213] provide a formal proof

Prover	Verifier
$S, Z \in QR_n$ $R_i \in QR_n \forall i \in [1, L-1]$ $e : l_e > l_m + 2$ $v : l_v = l_n + l_m + l_r$ $n = pq$ $A = (\frac{Z}{R_0^{m_0} \dots R_{L-1}^{m_{L-1}} S^v})^{1/e}$	
	$\xrightarrow{e, A, v}$
	$e \stackrel{?}{\in} [2^{l_e-1}, 2^{l_e}]$ $m_i \stackrel{?}{\in} \pm\{0, 1\}^{l_m} \forall i$ $Z \stackrel{?}{=} A^e R_0^{m_0} \dots R_{L-1}^{m_{L-1}} S^v$

Table 3.9: Camenisch-Lysyanskaya signature scheme. k denotes the security parameter, p, q are safe primes. QR is the group of quadratic residues. M denotes the set of messages $\{(m_0, \dots, m_{L-1}) : m_i \in \pm\{0, 1\}^{l_m}\}$. All operations are $\text{mod } n$.

of the security of protocol 3.9 against adaptive chosen-message attacks under the strong **RSA** assumption (see section 3.2.3).

The scheme in protocol 3.9 provides the core structure for building a selective disclosure system. In particular, it permits to build a signature protocol where the prover demonstrates possession of a signature by disclosing only a subset of the original messages. The construction refers again to the work proposed in [213], and incorporate the improvements presented in [219, 220]. Let us consider again the set of messages M . We partition M into three distinct subsets using the set of indexes I_h, I_c, I_r . These sets respectively indicate the indexes of hidden messages, committed messages and revealed messages chosen by the prover. We now observe that it is possible to generate valid signatures starting from the original signature (e, A, v) . In fact, the tuple $(A' = AS^{-r} \text{ mod } n, e, v' = v + er)$

with $r \in \{0, 1\}^{l_n + l_\emptyset}$ random value is a valid signature that correctly verifies the checks in protocol 3.9. The randomization of the signature permits to publish it without compromising the scheme security. Indeed, provided that $A \in \langle S \rangle$ and that r is chosen uniformly, the value A' is statistically close to uniform over $\mathbb{Z}_x^*$. By sharing the randomized A' , the verifier can now validate the signature with partial knowledge over the set of messages. The particular construction of the zero-knowledge proof of knowledge permits to the prover to demonstrate that

Prover	Verifier
$r, r' \in \{0, 1\}^{l_n + l_\emptyset}$ $A' = AS^r$ $C = g_0^{r'} \prod_{i \in I_c} g_i^{m_i}$	
$\xrightarrow{A', C, \{m_i\}_{i \in I_r}}$	
$PK\{(\varepsilon, \tilde{v}, \rho', \{\mu_i\}_{i \in I_c \cup I_b}) :$ $ZA'^{-2^{l_e-1}} \prod_{i \in I_r} R_i^{-m_i} = \pm A'^{\varepsilon} S^{\tilde{v}} \prod_{i \in I_c \cup I_b} R_i^{\mu_i} \wedge$ $C = g_0^{\rho'} \wedge$ $\mu_i \in \{0, 1\}^{l'_e + l_\emptyset + k + 1} \forall i \in I_c \cup I_b \wedge$ $\varepsilon \in \pm \{0, 1\}^{l'_e + l_\emptyset + k + 1}$	
$m_i \in \pm \{0, 1\}^{l'_m} \forall i \in I_r$	

Table 3.10: Camenisch-Lysyanskaya Selective Disclosure scheme. k denotes the security parameter. A, S, Z are calculated as in protocol 3.9. $G = \langle g_0 \rangle$ denotes a group of prime order $q > 2^{l_m}$. $g_1, \dots, g_L$ are elements of G . All operations are *modn*

$m_i \in 0, 1^{l'_m}$ and $e \in [2^{l_e-1} + 1, 2^{l_e} - 1]$. These requirements are crucial for guaranteeing the security of the protocol. It is clear from direct examination of protocol 3.10 that the verifier gets to know only the subset of messages $m_i : i \in I_r$. In other words, the prover can use the scheme to decide which data is publicly disclosed, thus enabling selective disclosure. It is noteworthy that protocol 3.10 implements simple disclosure of attributes. However, the prover can add more sophisticated disclosure techniques by applying the zero-knowledge techniques illustrated in section 3.2.4.

Boneh-Boyen-Shacham scheme. Despite its significant successes as key component Anoncred frameworks [215], the **CL** signature 3.2.5 depends on the **RSA** robustness assumption 3.2.7. For this reason, **CL** signatures require the use of long keys and signatures to achieve satisfactory to achieve levels of security, resulting in slow cryptographic operations. In contrast, the **BBS** signature places its trust in the **q-sDH** assumption 3.2.12 combined with pairings based on elliptic curves. This alternative approach requires keys and signatures significantly shorter while maintaining an equivalent level of security. **BBS** is a digital signature scheme categorized as a form of short group signature that supports several unique properties. Notably, the scheme supports signing multiple messages whilst producing a single output digital signature. Through this capability, the possessor of a signature is able to generate proofs that selectively disclose subsets of the originally signed set of messages, whilst preserving the verifiable authenticity and integrity of the messages. Furthermore, these proofs are said to be zero-knowledge in nature as they do not reveal the underlying signature; instead, what they reveal is a proof of knowledge of the undisclosed

signature.

In the **BBS** signature, G_1, G_2 are defined as groups of points over elliptic curves (see section 3.2.3). As such, they are written as additive groups. On the other hand, G_t is a finite field written as a multiplicative group. With such premises, the bilinear map for **BBS** can be formulated as

$$e(P_1, P_2) = e(a \cdot g_1, b \cdot g_2) = e(g_1, g_2)^{ab} = g_t^{ab} \quad (3.11)$$

with $P_1 \in G_1$ and $P_2 \in G_2$. The **BBS** signature protocol consist of three operations: key generation, signature generation, and signature verification. Given a set $M = \{m_1, \dots, m_L\} \in \mathbf{Z}_p$ of messages, the **BBS** signature produces a public key consisting of a tuple $PK \leftarrow (w, b_0, \dots, b_L) \in G_2 \times G_1^{L+1}$. The signature of the messages can then be defined as $\sigma \leftarrow (A, e, s) \in G_1 \times \mathbf{Z}_p^2$. Protocol 3.11 display the complete scheme. The last check operation holds

Prover	Verifier
$\{b_i\}_{i \in [0, L]} \xleftarrow{\text{random}} G_1^{L+1}$ $x \xleftarrow{\text{random}} \mathbf{Z}_p^*$ $w \leftarrow g_2^x$ $PK \leftarrow (w, b_0, \dots, b_L) \in G_2 \times G_1^{L+1}$	
	$\xrightarrow{PK}$
$(e, s) \xleftarrow{\text{random}} \mathbf{Z}_p^2$ $A \leftarrow (g_1 b_0 \prod_{i=1}^L b_i^{m_i})^{\frac{1}{e+x}}$ $\sigma \leftarrow (A, e, s) \in G_1 \times \mathbf{Z}_p^2$	
	$\xrightarrow{M, \sigma}$
	$e(A, w g_2^e) \stackrel{?}{=} e(g_1 b_0^s \prod_{i=1}^L b_i^{m_i}, g_2)$

Table 3.11: Boneh-Boyen-Shacham signature scheme.

because of the property of bilinear pairing: $e(A, w g_2^e) = e((g_1 b_0^s \prod_{i=1}^L b_i^{m_i})^{\frac{1}{e+x}}, g_2^{e+x}) = e(g_1 b_0^s \prod_{i=1}^L b_i^{m_i}, g_2)$. The prover can demonstrate knowledge of the signature without disclosing all the original messages. Let $\{m\}_{i \in \mathbf{D}}$ the set of disclosed messages, where $\mathbf{D}$ is the set of disclosed indexes. Similarly, we define $\{m\}_{i \notin \mathbf{D}}$ to be the set of hidden messages. Protocol 3.12 shows the selective disclosure scheme. The signature proof SPK consists of two sub-proofs concatenated with a logical AND $\wedge$. The first sub-proof proves the validity of the signature σ , while the second sub-proof proves the validity of the hidden attributes. The proof of knowledge corresponds to $(\hat{A}, \bar{A}, d, \pi)$. It is noteworthy that the proof reveals neither the signature σ nor any of its variables. Hence it can be safely sent to the verifier.

Zero-Knowledge Succinct Non-interactive Arguments of Knowledge A **zk-SNARK** represents a cryptographic protocol that allows a prover to demonstrate that have specific information about a process while maintaining confidentiality. The term **zk-SNARK** encompasses several key aspects and it is useful to explore each element individually

- **Zero-Knowledge.** This allows the prover to communicate certain information to the verifier while refraining from disclosing the information.

Verifier

Prover

$\{b_i\}_{i \in [0,L]} \xleftarrow{\text{random}} G_1^{L+1}$
 $x \xleftarrow{\text{random}} \mathbf{Z}_p^*$
 $w \leftarrow g_2^x$
 $PK \leftarrow (w, b_0, \dots, b_L) \in G_2 \times G_1^{L+1}$

$\xrightarrow{PK}$

$(e, s) \xleftarrow{\text{random}} \mathbf{Z}_p^2$
 $A \leftarrow (g_1 b_0 \prod_{i=1}^L b_i^{m_i})^{\frac{1}{e+x}}$
 $\sigma \leftarrow (A, e, s) \in G_1 \times \mathbf{Z}_p^2$
 $(r_1, r_2) \in \mathbf{Z}_p^* \times \mathbf{Z}_p$
 $r_3 \leftarrow \frac{1}{r_1}$
 $\hat{s} \leftarrow s - r_2 r_3$
 $\hat{A} \leftarrow A^{r_1}$
 $\bar{A} \leftarrow \hat{A}^x$
 $d \leftarrow (g_1 b_0^e \prod_{i=1}^L b_i^{m_i})^{r_1} b_0^{-r_2}$

$\xrightarrow{\{m_i\}_{i \in \mathbf{D}}, \hat{A}, \bar{A}, d}$

$SPK\{(\{m\}_{i \notin \mathbf{D}}, e, r_2, r_3, \hat{s}) :$
 $\quad \bar{A}/d = \hat{A}^{-e} b_0^{r_2} \wedge$
 $\quad g_1 \prod_{i \in \mathbf{D}} b_i^{m_i} = d^{r_3} b_0^{-\hat{s}} \prod_{i \notin \mathbf{D}} b_i^{-m_i}\}$

$\bar{A} \stackrel{?}{\neq} 1_{G_1}$
 $e(\hat{A}, X) \stackrel{?}{=} e(\bar{A}, g_2)$

Table 3.12: Boneh-Boyen-Shacham selective disclosure scheme.

- **Succinct.** The conciseness of the **zk-SNARK** ensures that the verification can take place quickly, often within a few milliseconds, since the test length usually extends only to a few hundred bytes at most.
- **Non-Interactive.** **zk-SNARK** are non-interactive in nature, involving a single message from the demonstrator to the verifier to establish the evidence.
- **Argument.** The term argument is used to describe these proofs, as they deviate somewhat from the conventional definition of proofs but effectively fulfil the same purpose.
- **Knowledge.** It refers to the information held by the demonstrator.

In the context of digital identities, **zk-SNARKs** serve as a mechanism to provide an efficient selective disclosure method in a verifiable credentials context. The zk-SNARK term itself was introduced in [221], building on [222] with following Pinocchio protocol [223, 224] making it applicable for general computing.

The efficiency of the method is rooted in the concept of polynomials, which can be represented as curves on a graph and are expressed in the form $c_n x^n + \dots + c_1 x + c_0$. The rank of a polynomial, indicated by its exponent higher than x (*i.e.*, n), plays a fundamental role. An important property of polynomials is that when it comes to two distinct polynomials of degree at most d , their intersection can occur in no more than d points. Basically, it is impossible to find two different polynomials that share a consecutive segment of a curve. To determine the intersections of two polynomials, they must be equated. The result of such an equation for polynomials of arbitrary grade d is always another polynomial with a degree not exceeding d . This is because there is no multiplication operation that can generate higher grades. The Fundamental Theorem of Algebra reinforces this concept by stating that a polynomial of degree d can have at most d solutions and, consequently, at most d common points. In the context of this method, knowing a polynomial implies that the prover has knowledge of its coefficients $c_{i \in [0, n]}$. Applying the Fundamental Theorem of Algebra, any polynomial can be broken down into the form $(x - a_0)(x - a_1) \dots (x - a_n) = 0$, where $a_{i \in [0, n]}$ are called the roots of the polynomial, representing its solutions. The prover's goal is to prove that he knows the roots $t(x)$ of the polynomial $p(x)$. This can be achieved by expressing $p(x)$ as the product of a target polynomial (known to the prover) and an arbitrary polynomial: $p(x) = t(x)b(x)$. To determine $b(x)$, the prover can use polynomial division, resulting in $b(x) = \frac{p(x)}{t(x)}$. Importantly, this protocol requires only a single round and provides a high level of confidence in the statement, assuming d is significantly smaller than the upper limit of the x range.

zk-SNARK uses homomorphic encryption to impose restrictions on the prover's x . This encryption technique allows the encryption of a value, while permitting the application of arithmetic operations to the encrypted data. A direct approach employs modular exponentiation, where a g base and a n module are chosen. The encryption of x takes the form of $X = g^x \mod n$. The multiplication of encrypted numbers can be obtained by calculating $X' = (g^x)^y \mod n = g^{xy} \mod n$. Similarly, the addition occurs as follows: $X \cdot Y = (g^x \cdot g^y) \mod n = g^{x+y} \mod n$. Subtraction and division naturally follow as reverse operations. We can explicitly define the encryption function as $E(x) = g^x \mod n$. There are limitations within this homomorphic encryption scheme. While it is possible to multiply a cipher value by an unencrypted one, multiplication (and division) of two ciphertexts, as well as exponentiation of a ciphertext value, are not feasible. These properties form the security basis of **zk-SNARK**. Knowledge of a polynomial involves the knowledge of its coefficients $c_{i \in [0, d]}$. With homomorphic encryption, it is conceivable to limit the prover's choice of encrypted powers, although such a restriction is not enforced. To satisfy the equation $z_p = (z_b)t(r)$, the demonstrator could, for example, calculate $z_b = g^r \mod n$ and $z_p = (g^{t(r)})^r \mod n$, where r takes a random value and $g^{t(r)}$ can be derived from the values provided by the verifier. The Knowledge of the Exponent assumption, as discussed in [225], enforces the usage of provided encrypted powers by introducing shifted versions of the powers. The prover is then forced to perform the operation on both versions of powers, thus enabling a control over them. Indeed, such protocol provides a proof that the prover did not use any other operation (*e.g.*, addition, multiplication) since it would erase the shift relationship. Prover eventually adds zero-knowledge by shifting the elements of a random value δ . The protocol can be made non-interactive by employing cryptographic pairings over the public parameters. Here they assume particular importance because the results cannot be used in other pairing operations. Protocol 3.13 summarizes the scheme. Here the polynomial $t(x)$ and the degree d must be agreed before execution. In the context of verifiable credentials, the attributes can be encoded as values of the polynomial. At that point, the scheme in protocol 3.13 enables to construct a selective disclosure system.

Table 3.13: Zero-knowledge proof of knowledge of a single polynomial. PK denotes the proving key, while VK the verification key.

4

Toward a Dynamic Identity Framework for IoT

In Chapter 2 we introduced cryptographic secrets as a convenient and secure way for identifying entities in the digital world. However, using a single identifier lacks in capturing the complexities and dynamics of the interconnected world. In Chapter 3 we proposed identities formed by multiple attributes to offer a more comprehensive approach. Nevertheless, such representation requires considerable efforts to maintain the intricate hierarchical structures (as we will see in Section 4.2).

This chapter undertakes a search for a more adaptable and expressive framework, capable of accommodating the complexity of modern identities. This framework aims not only to label entities, but to encompass their multiple characteristics and relationships in a flexible, scalable and understandable way. Section 4.1 begins with an analysis of the technologies proposed in the academic domain for the representation of knowledge. Their potential, limitations and applicability in the context of IoT identity frameworks are examined. Section 4.2 deepens the challenge of conceptual organization, drawing on the teachings of the semantic web and its implications for the representation of identity. Section 4.2 attempts to address these complexities by proposing recent advances in **LLMs** emerge as a beacon of possibilities. The section examines their potential role in building a flexible and robust framework for the representation of identities. To further support the analysis, Section 4.4 evaluates the practical capabilities of a prominent language model, ChatGPT, in the representation of information. Through an analysis of its performance in code debugging, a task that requires a nuanced understanding and representation of intricate data, the section aims to assess the actual scope and limitations of such models in the field of identity representation.

4.1 TECHNOLOGIES FOR REPRESENTING KNOWLEDGE

The traditional method of identity modeling (see chapter 2) is a direct and efficient approach to managing and solving data friction between physical and virtual world. However, it lacks the ability to represent the hierarchy and relationships of data or manage complex queries. In contrast, it is possible to create meaningful content by employing more sophisticated modeling methodologies. At a lower level, markup languages (*e.g.*, **XML**, **JSON**) offer a standardized way to describe and exchange structured data, regardless of the application. One example is SensorML [226], which provides a standardized model and **XML** encoding to describe various sensors, improving interoperability and facilitating their use in complex workflows. In addition, there is **UML**, an open method for explaining, visualizing and building complex systems, particularly in software architecture. **UML** incorporates engineering best practices, with examples such as the Prehofer's **UML** models for complex **IoT** systems [227].

The Semantic Web [228] combines the data and demand of the **IoT**, offering a versatile modeling method through ontology. Ontology defines classes, instances and attributes, enriching data as knowledge through a shared language [229]. **RDF** [230] offers a unified standard for describing Web resources (*e.g.*, classes, properties, instances) in a triple subject-predicate-object format. For efficient transmission, **RDF** can be serialized with different methodologies: **RDF/XML**, N-Triples, Turtle, **RDFa**, **JSON-LD**. Hasemann et al. [231] introduced HDT Streaming for efficient **RDF** data transfer, while Gutierrez et al. [232] incorporated time into **RDF** reasoning [233].

RDF, used for resource relationships, lacks definitions of hierarchical class relationships. **RDFS** [234] improves this by extending **RDF** vocabulary, allowing users to standardize domain classes and attributes within **RDF** descriptions. To address the limitations of **RDF**, the W₃C introduced the **OWL** in 2004 [235]. **OWL** enriches

resource descriptions with better semantic expression. **OWL** aims to balance the expressiveness of language and intelligent services such as reasoning. It includes three increasingly expressive sub-languages: **OWL Lite**, **OWL DL** and **OWL Full**. In 2012, the W₃C published the technical specifications of **OWL2** [236], presenting the most expressive SROIQ description logic, albeit with greater computational complexity. **OWL2** introduces new features, syntax, and profiles, relaxing some restrictions on **OWL DL** [237].

Additionally, the W₃C has proposed two ontological languages specifically designed for sensors and actuators. The **SSN** defines a sensor-related ontology, which includes variables, procedures, study characteristics, samples, observed properties, and execution mechanisms [238]. The **SAN** introduces concepts such as actions, execution devices, features, scope and more for actuators [239]. The **IoT-O** ontology, a central **IoT** domain ontology, expands with application-specific knowledge, incorporating **SSN**, **SAN**, and other modules [229]. Using identity modeling techniques, we improve the construction of the semantic Web, making data conform to specifications easily understood by computers. This facilitates the smooth sharing and reuse of data between different applications and systems. The semantic web-based architecture creates a knowledge-relationship graph, effectively addressing the information overload and enabling the implicit discovery of relationships. Ganz et al. [240] introduced a method for acquiring knowledge, automatically extracting rules and generating an ontology of themes from sensor data. The fusion of this technology with the research on the **IoT** identity modeling promises interesting research opportunities.

4.2 THE PROBLEM OF CONCEPTUAL ORGANIZATION: THE SEMANTIC WEB CASE

OWLs represent a significant learning challenge [241, 242]. They require a change of perspective, especially for individuals with experience in software engineering, object-oriented programming, or relational databases. Those who are new in this field are faced with the arduous task of creating an innovative model adapted to their specific problem or domain. In addition, they must confront themselves with the search for suitable tools, and establish effective workflows and development processes. Unfortunately, the resources to guide informed decision-making in this context are limited. Many people entering the world of semantic technologies initially choose simpler modeling languages like **SKOS** [243] to define their vocabularies. However, they often encounter difficulties in trying to transfer their models to more expressive languages, as there is no direct path between them [244]. Many application developers begin their work with alternative representations (*e.g.*, **UML** diagrams, mind maps, **XML** schemas or spreadsheets). These representations require conversion to the **OWL** format. Currently available conversion algorithms perform mainly structural transformations driven by predefined mapping rules. The results of these conversions often require manual refinement to accurately capture the predicted semantics of the source data. Although the beginning of ontologies is of significant importance in the process of developing ontologies, acting as an initial obstacle for beginners in the field, it has not received as much attention in research as other related aspects. In addition, robust methodological approaches and appropriate tool support are lacking.

The intrinsic complexity of **OWL** further complicates the creation of developer friendly **APIs** to interact with **OWL** ontologies. The most comprehensive Java **API** for **OWL 2.0** ontologies, known as **OWLAPI** [245], requires a solid understanding of **OWL** specifications and can be intimidating for developers. Numerous academic

and commercial initiatives have already used GraphQL to interrogate **RDF** graphs. However, it is clear that further research is imperative to address the complexities of seamless integration between GraphQL, **RDF** and SPARQL [246]. The adoption of ontologies encounters obstacles due to the existence of alternative approaches more user-friendly than **OWL**. In the 2013 Common Crawl dataset, Microdata and microformats were significantly more common than **RDFa** [247]. Schema.org, for practical reasons, opted for an extension of the **RDFS** schema as its data model* instead of **OWL**. It introduced *domainIncludes* and *rangeIncludes* as substitutes for *rdfs:domain* and *rdfs:range* to facilitate encoding of multiple values for domains and property ranges. ShACL and ShEx [248], designed to validate graph-based data against a set of conditions, are perceived by some within the community as simpler knowledge representation languages that could potentially replace OWL in the future. In the commercial sector, labelled property graphs are gaining support for encoding knowledge graphs. This preference stems from the perception that they are easier to understand and generally offer higher query performance.

Ontology engineering includes a crucial element of reuse, applicable both to entire ontologies and to specific components within them. Despite the existence of numerous ontology repositories [249] (e.g., BioPortal, AgroPortal, Linked Open Vocabularies, Ontohub), the task of identifying the most suitable ontology for a specific purpose remains challenging. A common scenario involves the search for an ontology suitable for the annotation of a corpus of texts. In the biomedical domain, the BioPortal Ontology Recommender uses various criteria to suggest appropriate ontologies. However, a comparable general recommender is currently lacking. A significant obstacle in the search for the right ontology is the development of a complete set of metrics capable of evaluating the adequacy of an ontology for a given task or domain [250, 251]. Using segments of an ontology, whether they are individual entities, subtrees, or collections of axioms, is often imperative during the process of developing ontology. However, implementing such re-use in practice presents considerable challenges. Numerous research studies have shown that the degree of ontological reuse remains remarkably low [252, 253]. By way of illustration, a recent survey by Kamdar et al. [253] revealed that in the field of biomedical ontologies, effective reuse of time limits amounts to less than 9%, despite a significant overlap of terms between 25% and 31% between them. Surprisingly, most ontologies tend to reuse less than 5% of their terms, drawing mainly from a limited selection of widely recognized ontologies. Challenges associated with achieving effective reuse cover a broad spectrum, including identifying suitable ontology for reuse, extracting relevant, and ensure the continued compatibility of the extracted subset while the source ontology undergoes evolution [254].

One concrete example showcasing these challenges is the development of the Semantic Web. The beginning of the Semantic Web can arguably be traced back to the year 1998 when the first iterations of the **RDF** [234] and the **RDFS** [255] specifications were introduced. Now, after more than two decades, there are varying points of view regarding the degree to which the original vision was realized, or even the feasibility and desirability of its complete realization. In the Semantic Web community, there has long been a consensus that, although the complete realization of the vision has yet to be achieved, it is only a matter of time, not a question of whether it will happen. In 2006, Shadbolt et al. [228], recognizing that the Semantic Web had not yet gained substantial traction, expressed the belief that it would soon become a significant presence once the relevant standards were solidly established. In 2007, Horrocks [256], similarly recognizing that the complete realization of the Semantic Web still seemed to be at some distance, argued that OWL had already achieved considerable success and had become the de facto standard for the development of ontologies in a wide range of fields, from geography and geology to

*<https://schema.org/docs/datamodel.html>

astronomy, agriculture, defense and life sciences. The following years were characterized by a widespread sense of optimism about Linked Data, with various authors claiming a dramatic increase in the publication of data in accordance with the principles of Semantic Web [257, 258]. In addition, this optimism has been reinforced by the fact that leading organisations, including Oracle [259], Facebook [260], and Amazon [261] have selectively embraced Semantic Web technologies, demonstrating increasing acceptance and support for these principles.

Despite the Semantic Web has met strenuous supporters over the years, there has been a growing contingent of critics. The cornerstone of the Semantic Web lies in the concept that machines will independently process structured content on the web. However, this process is highly susceptible to fragility when confronted with accidental errors and deliberate deception, often perpetuated by unreliable publishers, which are common on the web. The vulnerability of the Semantic Web to ontological disagreements is another critical point of discussion [262]. In reality, there is no single universally accepted way to model a domain using an ontology, for there is no universal truth. Different stakeholders in a domain can attribute different meanings to terms or even support contradictory statements, making the Semantic Web fragile in the face of different perspectives. Another criticism of the Semantic Web is its perception of lack of scalability [263]. Consuming data published using Semantic Web standards requires algorithms with inadequate scalability and performance. Existing implementations often show poor performance and scalability metrics, further undermining the feasibility of the Semantic Web. Moreover, there is the argument that advances in Machine Learning will make the Semantic Web obsolete before it can gain significant diffusion [264]. The Semantic Web premise is based on the assumption that the current HTML-based web is inadequately readable by machines. However, the continuous progress of Machine Learning is gradually dismantling this assumption. As the Semantic Web develops and potentially exerts considerable influence on the web, Machine Learning technologies will have evolved to the point where these technologies of publishing and consuming structured content will become redundant.

4.3 LARGE LANGUAGE MODELS FOR FLEXIBLE IDENTITIES

LLMs have had a profound impact in the field of **KR** and have significantly influenced the wider world. They have demonstrated human level performance in a wide range of natural language tasks. As a result, there is a growing acceptance of the idea that knowledge can be effectively encoded within the parameters of certain linguistic models. This represents a significant milestone for the field of **KR**. Historically, the focus had been mainly on explicit knowledge, whether it be incorporated in textual form or structured in knowledge databases and graphs [265]. With the advent of the **LLMs**, a crucial turning point was reached, marking a shift from the predominant emphasis on the explicit representation of knowledge to a renewed focus on hybrid representation, which includes both explicit and parametric knowledge. The **KGs**, a well-established method for explicit representation of knowledge, are now the subject of extensive surveys for integration with transformer-based **LLMs**, including BERT [266], RoBERTa [140], as well as the more recent generative **LLMs** such as the GPT series [267] and LLaMa [268]. Some research efforts use **LLMs** to improve **KGs** through activities such as knowledge mining, **KG** construction and refinement, while others employ **KG** to improve **LLMs** through activities such as prompt training and learning or knowledge enhancement.

The simultaneous use of parametric and explicit knowledge is the subject of ongoing debate in the community, with supporters and skeptics presenting divergent views. **KGs** offer a structured means of representing knowledge

with explicit relationships, facilitating reasoning and inference [269]. Detractors argue that parametric knowledge within **LLMs** is based primarily on statistical models rather than genuine understanding and reasoning [270]. Supporters of **LLMs**, such as ChatGPT, emphasize their ability to generalize from extended corpora of text, capturing a wide spectrum of information and demonstrating exceptional language comprehension skills. On the other hand, **LLMs** may generate plausible but erroneous or meaningless answers, including hallucinations, due to the absence of an explicit representation of knowledge [271]. There are also doubts as to whether **LLMs** could include directional implications [272] or infer inclusion relationships between concepts [273]. In contrast, **KGs** may require considerable resources to be built. Training **LLMs** can also involve significant costs, but they can easily be used to support many subsequent applications, moving **AI** from a background role to a central one. Consequently, it is clear that parametric knowledge is not the sole objective of **LLMs**.

The success of **KGs** largely depends on their ability to provide highly accurate factual information about entities. For example, YAGO claims a precision rate of over 95% [274]. Similarly, Google requires high accuracy in its **KG** for operational purposes, as demonstrated by the non-use of Knowledge Vault’s semi-automatic construction method, partly because of its failure to reach the desired 99% accuracy benchmark [275]. In this context, many **LLM**-based approaches to **KG** completion struggle to achieve these high levels of performance, as demonstrated by the performance of BERT in [276], and of GPT-3, equipped with hundreds of billions of parameters [277]. There are however several interesting properties of **LLMs** in this context. Knowledge has traditionally been transmitted through written texts. Consequently, a substantial part of contemporary knowledge exists in textual format. By leveraging **LLMs**, quick access to extensive textual corpora is facilitated, including on standard consumer hardware. This ability allows **AI** developers to overcome the challenges of large-scale data collection, preparation, storage, and query. It also reduces the dependence on the information retrieval field. While the most obvious competences of **LLMs**, such as answering questions and dialogue, continue to be subject to critical evaluation, it is imperative not to neglect their substantial contribution to simplifying many conventional knowledge engineering activities. The **LLMs** can excel in various activities, including dependence and structured analysis, entity recognition and relationship extraction, with refinement through some specimens, and through a prompting approach with just a few strokes. Errors and improvements propagate through the process, facilitating the construction of **KGs** on an unprecedented scale and with improved quality. In addition, **LLMs** find utility in a multitude of subsequent activities outside the field of knowledge engineering. By providing explicit, in particular structured, knowledge in **LLMs**, for example through research improvement methods, it is possible to make explicit knowledge more easily applicable in a wide range of subsequent activities. **LLMs** have already made significant progress in increasing understanding of natural language, as demonstrated by their performance in tasks such as textual implication, the synthesis, detection and generation of paraphrases. These skills play a crucial role in consolidating knowledge engineering against language variations, typographical errors, redundancy, and other idiosyncrasies found in human-written texts from the web and other noisy forms of text. The integration of parametric knowledge with explicit knowledge opens the possibility of achieving an even more advanced understanding of language, not only in the field of textual implication, but also in other tasks of **NLP** such as synthesis and consistent content generation. In traditional knowledge engineering, a crucial phase involves the consolidation and aggregation of contradictory and agreed information. This requires complex methods to reconcile observations from sentences, patterns and constraints [278]. During the formation of **LLMs**, this aggregation process occurs automatically. Although not fully understood, this phenomenon has the potential to alleviate a significant challenge in knowledge engineering, potentially outsourcing the task of consolidating diverse information.

The analysis provided introduces the relationship between digital identities and their representation. The concept of connecting identity and representing knowledge explores the intricate relationship between how we define our identities and how we represent the knowledge associated with those identities. Our identities are sets of properties that can change over time as they are shaped by the accumulated information and experiences (see section 3). As a result, their representation is highly subjective and requires a flexible solution. One way to represent knowledge is through a concept graph, in which each concept represents a property or attribute related to a person's identity. This graph can include information such as age, gender, education, interests, and more. These properties collectively form a person's identity, and the connections between them define the unique characteristics of the individual. However, it is essential to recognize that these sets of concepts that make up an identity are context-related. Depending on the situation or perspective of the observer, different aspects of a person's identity may emerge. For example, in a professional context, a person's educational background and professional achievements may be more relevant, while in a social context, their hobbies and interests may take precedence. The challenge here is that the representation of identity inherits the limits of the representation of knowledge. To address this problem, one can consider the potential of **LLMs** as a flexible solution to represent digital identities. **LLMs**, like GPT-3.5, offer a powerful tool for representing knowledge. They can manage relatively complex queries and provide context-dependent representations based on the input they receive. This flexibility makes them suitable to capture the dynamic nature of identities. In a possible framework for flexible identities, public **LLMs** can be exploited to provide representations of queries related to an individual's identity. For example, queries such as *Man, 31 years old, graduate, Male graduated of thirty-one years* and *Un homme diplômé de trente et un ans* can be used to construct a complete representation of an individual's identity. The **LLM** would output a sequence of numbers representing the particular context elicited by the input query. Despite the three examples use different syntax and languages, the output sequence will have similar values. This is the main property of **LLMs**, as they are capable of capturing the semantic of the queries, thus providing an useful tool for comparing concept similarity. To provide security, this representation can be signed as a verifiable credential (see section 3.2), ensuring its authenticity. The signed context can subsequently be used to selectively prove some credentials. A verifier first check the validity of the signature. He then inputs the sequence as context to the **LLM** and ask questions to it. For examples, he can ask whether the man is older than 18 or not, or if he is graduated. The **LLM** would then answer accordingly, thus providing a rudimentary selective disclosure system based on large language models.

The approach just illustrated allows a fluid and adaptable representation of digital identities. To provide a thorough analysis, section 4.4 shows a critical examination over the capabilities of ChatGPT of representing knowledge. In particular, it assesses its knowledge representation performances by examining the code vulnerability detection rate.

4.4 CHATGPT'S ROLE IN CYBERSECURITY: A CRITICAL EXAMINATION OF CODE VULNERABILITY DETECTION

AI shows considerable potential in rationalizing various complex software engineering tasks. These tasks include code generation [279, 280], synthesis/explanation [281, 282], and program correction [283, 284]. Currently, cutting-edge code generation solutions leverage **AI** models to independently create complete or partial software

programs based on natural language descriptions or code input. To address code synthesis, researchers use generative AI models that can produce natural language descriptions when code input is provided, with the goal of improving code understanding, maintainability, and reusability. In the field of program repair, neural network-based models have emerged as the standard for learning how to generate corrections, thus reducing the need for manual debugging and improving software reliability and security.

The recent rise of **LLMs** has attracted considerable attention, not only in society at large, but also in the field of software engineering [285, 266, 286]. These **LLMs**, pre-defined on vast collections of data containing source code and text in natural language, have demonstrated remarkable competence in understanding code structures and in generating code or textual content. Progress led by **LLMs** has significantly enhanced the effectiveness of automated techniques applied to different software engineering challenges, including program repair [287, 288]. A prominent example of **LLM** is OpenAI Codex [285], the underlying model of Github Copilot [289]. Codex has demonstrated its competence by accurately and efficiently generating code snippets [290], identifying and correcting software bugs or vulnerabilities [291] and providing concise summaries for code segments [292]. Although the experimental results of the literature highlight the immense potential of **LLMs** in the software development community, the overall performance achieved so far remains somewhat limited [293, 294].

ChatGPT [295], a newly introduced **LLM**, has aroused considerable interest in the field of software engineering due to its capabilities reported in automating software repair with minimal human intervention, a long-term aspiration in the field [296, 295]. Early findings suggest that ChatGPT has the potential to be transformative and the future of **LLM**-based software engineering looks promising. However, it is essential to conduct further investigations to fully assess the capabilities of **LLMs**. Existing studies on repairing programs with ChatGPT have tended to evaluate its performance using publicly available benchmark data that may have affected ChatGPT's training corpus, thus limiting the generality of the results reported to new and unforeseen problems.

To address these gaps, we propose an empirical study to assess the actual potential of ChatGPT as an automated programming assistant. One notable strength that ChatGPT has shown is its ability to detect vulnerabilities. However, this ability has raised questions about model training data. It remains unclear whether the databases were included in the training corpus, which is a crucial factor that could significantly affect the performance of the model. The absence of this information casts a shadow of doubt on current analyses, as the composition of the model training data is a fundamental determinant of its capabilities. To address this uncertainty and more accurately assess ChatGPT's real capabilities, we adopted refactoring techniques to the source code, while preserving its semantic meaning. In doing so, we aimed to measure the model's ability to understand and manipulate code beyond mere storage. However, the results of these experiments showed a worrying trend: a decrease in performance as refactoring levels increase (from 54% to 34%). This suggests that ChatGPT could rely on the mechanical storage of training data instead of the actual understanding of the underlying logic of the code. In addition to refactoring, we explored more sophisticated prompting techniques to unlock higher reasoning capabilities in ChatGPT (*i.e.*, Tree of Thought technique [297]). Although this approach shows some promise in pushing the model towards a more human-like reasoning, the progress achieved is modest at best. It seems that prompting may push ChatGPT towards more advanced reasoning, particularly in the context of vulnerability detection, but it may not be a panacea for its limitations.

4.4.1 RELATED WORKS

LLMs have exerted a profound influence in the field of **NLP** in recent years, demonstrating their versatility in different fields [298, 299]. Among these models, the encoder model known as **BERT** [266] has gained wide recognition. **BERT** is a transformer-based architecture and uses a Masked Language Modeling pre-training task, in which it hides certain input tokens and tries to predict them from the remaining tokens bidirectionally. This approach allows **BERT** to capture contextual representations of words and intricate relationships between words within sentences. Consequently, **BERT** has been widely used as an embedding model (*i.e.*, encoder) for subsequent software engineering tasks [300, 301].

In contrast, the decoder model called **GPT** [285, 302] was trained by an unsupervised learning technique. **GPT** uses a unidirectional causal attention mechanism, allowing each position in the sequence to consider only the previous positions while preserving the input order. During pre-training, the model aims to maximize the probability of predicting a word based on its contextual clues. **GPT-1** [303] tries to predict the next word in a sequence by considering the previous words, a one-way task that allows it to learn the dependency relationships between words and generate contextually relevant content. **GPT-2** [304], on the other hand, exceeds **GPT-1** in scale, boasting 1.5 billion parameters, a tenfold increase over the 117 million parameters of **GPT-1**. **GPT-2** has been trained with a larger architecture on a more extensive and diverse dataset, which includes web pages, books and scientific articles. This has shown that training on a larger dataset and using more parameters can significantly improve the skills of a language model, even in zero-shot scenarios. Compared to its predecessors, **GPT-3** [267] marks a significant milestone in the evolution of **LLMs** as it uses 175 billion parameters. Codex [285], initially constructed using **GPT-3** training weights on a natural language corpus, was subsequently refined on an extensive dataset of code files. Subsequently, InstructGPT [302] was developed to be further refined on **GPT-3**, using human feedback-based reinforcement learning.

APR is a methodology developed to automatically detect and correct defects or vulnerabilities in software code [305, 306]. To illustrate this, Gulwani and colleagues [307] introduced an **APR** algorithm specifically designed for introductory programming exercises, taking advantage of students' correct solutions to correct incorrect attempts. Hu et al. [308] proposed a fully automated research-based approach that generates real-time repairs for students' programs by remastering the correct solutions and aligning the wrong programs with the remastered ones. This approach requires fewer accurate solutions and produces better results even with only one correct reference solution. From another perspective, systems based on **AI** have recently achieved remarkable performance in this field. Chakraborty et al. [309] introduced CODIT, a new tree-based neural network system, to learn and suggest code modification patterns from extended open source data. Its evaluation has proven its effectiveness in proposing corrections and correcting errors in the Defects4J dataset. Ye et al. [310] submitted RewardRepair, a program repair model that incorporates program-specific information during neural network weight optimization to generate corrections that are successfully compiled and are less susceptible to overfitting.

Pearce et al. [311] performed a large scale study of five commercially available, black-box, off-the-shelf **LLMs**, as well as an open-source model and locally-trained model, on a mix of synthetic, hand-crafted, and real-world security bug scenarios. The experiments demonstrated that while the approach has promise (the **LLMs** could collectively repair 100% of synthetically generated and hand-crafted scenarios), a qualitative evaluation of the model's performance over a corpus of historical real-world examples highlights challenges in generating functionally correct code.

Prenner et al. [312] used the multi-language QuixBugs benchmark (40 bugs in both Python and Java) and found that, despite not being trained for **APR**, Codex is surprisingly effective, and competitive with recent state of the art techniques. Their results also show that Codex is more successful at repairing Python than Java, fixing 50% more bugs in Python.

Sobania et al. [296] evaluate ChatGPT on the standard bug fixing benchmark set QuixBugs and compare the performance with the results of several other approaches reported in the literature. They found that ChatGPT’s bug fixing performance is competitive to the common deep learning approaches CoCoNut and Codex and notably better than the results reported for the standard program repair approaches. In contrast to previous approaches, ChatGPT offers a dialogue system through which further information can be entered (*e.g.*, the expected output for a certain input). By providing such hints to ChatGPT, its success rate can be further increased, fixing 31 out of 40 bugs, outperforming state-of-the-art.

Xia and Zhang [313] propose ChatRepair, the first fully automated conversation-driven **APR** approach that interleaves patch generation with instant feedback to perform **APR** in a conversational style. ChatRepair first feeds the **LLM** with relevant test failure information to start with, and then learns from both failures and successes of earlier patching attempts of the same bug for more powerful **APR**. For earlier patches that failed to pass all tests, they combine the incorrect patches with their corresponding relevant test failure information to construct a new prompt for the **LLM** to generate the next patch. In this way, they can avoid making the same mistakes. While their approach is general, they implement ChatRepair using ChatGPT and successfully fixed 162 out of 337 bugs for 0.42 each.

4.4.2 MOTIVATION

The results and methods in the context of **LLM** code repair discussed in section 4.4.1 research articles raise important points that are worth of discussion. Standardized databases, such as QuixBugs, are often used to evaluate the performance of **APRs** in code repair. These databases consist of publicly available code repositories with known bugs. For a fair evaluation of the **LLM**, it is important to consider whether these public repositories may have been included in the **LLM** training data. If they were included, it could potentially affect the results, as models may have already seen similar code and bug patterns during training. Dataset scripts often include comments that provide details about bugs, the context of the code, and possible corrections. An important question is whether these comments were removed or included during the **LLM** evaluation. This observation is significant because the comments in the code may contain valuable contextual information that could help in fixing bugs. If these comments were included in requests made to **LLMs**, it could explain some of the success in bug repair, especially for synthetic or craft scenarios. Earlier studies often used relatively simple requests to initiate bug repair tasks with **LLMs**. Another observation concerns whether the use of more sophisticated techniques in the construction of requests can lead to improved results. This observation is related to the question asked in [312] about the types of requests that produce the best results in using Codex for bug repair. The use of more nuanced and context-rich requests could be a potential way to improve the effectiveness of **LLMs** in repairing real-world bugs, as they could better capture code and bug complexities.

To further underline the concerns over the actual performances of ChatGPT as a code assistant, we designed a simple problem that contains a non-trivial answer. Consider the C code provide in algorithm 4.1.

The functions *HelloWorldChar()* and *HelloWorldFloat()* performs the same operation, that is outputting

Algorithm 4.1 Hello world in C. Function *HelloWorldChar()* directly put the string as an array of *char*. Function *HelloWorldFloat()* stores the string as an array of *float* and then casts it before printing.

```
#include <stdio.h>

int HelloWorldChar() {
    char b[] = "Hello World!";
    puts(b);
    return 0;
}

int HelloWorldFloat() {
    float b[] = {
        1.1431391224375825e+27,
        6.6578486920496456e+28,
        7.7392930965627798e-19,
        3.2512161851627752e-9
    };
    puts((char*)b);
    return 0;
}
```

the string *Hello World!* to the console. The difference lies in the type chosen for the array storing the string of characters. In other words, the two functions share the same semantic while having different syntax. The first function uses an array of *chars*, while the second one uses an array of *floats*. Ideally, ChatGPT should be capable of determining the output of these two functions. However, the results are quite limited and open to the themes of this work. When provided with the function *HelloWorldChar()*, the bot answers:

The given C code will print the string "Hello World!" to the console. [...]

When provided instead with the function *HelloWorldFloat()*, the bot outputs the following:

The code you provided attempts to print the contents of a float array b as if it were a character array (char). This is not a valid operation in C, and the output of this code is undefined. [...] Converting a float array to a char* and then using puts to print it will not give you meaningful results. [...]*

These results can be obtained with both ChatGPT_{3.5} and ChatGPT₄. They clearly showcase the limited reasoning capabilities of the bot. Despite the simplicity of the code base provided, the **LLM** is not capable of untangling the second function. At first impression, it seems that the algorithm lacks higher thinking capabilities, *i.e.*, recognizing the array of *floats* as a *char* array with valid characters requires more sophisticated reasoning.

From these considerations, we derive a list of research questions that will guide the design of our experiments.

- **Were the standard datasets included during the training phase?** We strive to determine if the public datasets were included during the training phase of ChatGPT. An affirmative answer would inevitably

deteriorate the reliability of previous works.

- **Were the script properly anonymized before experiments?** Standard datasets often contain inline comments with useful information for locating the bugs. Not removing such comments would completely invalidate the results.
- **What are the reasoning limits of ChatGPT?** ChatGPT must be evaluated on novel samples to verify its reliability in practice. It would also be useful to determine in which classes of bugs ChatGPT have a harder time in spotting.

4.4.3 METHODOLOGY

We adopted ChatGPT 3.5 in all of the experiments. We opted for the older version to establish a benchmark that could be used in future for baseline comparison. Despite criticisms around **GPT-4**'s stability and accuracy issues, as cited in multiple studies [314, 315, 316, 317], **GPT-3.5** has proven to be more dependable. Consequently, our research is focused on scrutinizing the default **GPT-3.5** settings on the ChatGPT web interface[†]. OpenAI's ChatGPT **API** lagged in performance when compared to its web interface, as highlighted by various reports. As a result, we opted to conduct all our experiments using the web interface for a more reliable evaluation.

We chose the Juliet dataset [318] as reference dataset for our experiments. Juliet is a NIST standard collection for vulnerability evaluation with test cases in the C/C++ language. It contains examples organized under 118 different **CWEs**, of which 108 in C language. The choice was driven by the well-documented repository, the quality of the examples and the extensive number of bug categories represented.

To determine if the dataset were included during the training, we opted for a simple prompt asking directly to the language model:

Do you know the JULIET (C/C++) dataset? Can you provide a sample code out of it?

We decided to work on three, progressive refactoring levels of the code. Using different levels permits to measure the performance of the model on different version of the same vulnerability. Ideally, ChatGPT should be independent of the wording of an algorithm and be capable of spotting a vulnerability regardless.

- **Level 0.** All the comments are stripped out for the source, all the conditionals are solved, and the defines are substituted into the code;
- **Level 1.** : Randomization of function and variable names;
- **Level 2.** : Different code base with the same vulnerability.

Level 0 produces a baseline code stripped of all the un-necessary information. Level 1 provides the first real level of re-factoring by randomizing the names of functions and variables. Level 2 provides an extreme version of re-factored code where only the semantic of the vulnerability is maintained. Differences in performances in the three levels could be a further indication of dataset inclusion during the training.

To further evaluate the capabilities, we integrated two different prompting styles. The simple prompt only asks to the model to spot the vulnerabilities.

[†]<https://chat.openai.com/chat>

Find all security vulnerabilities in the following code. Describe them in one phrase. The code snippet is:

The complex prompt instead uses a technique named Tree of Thought that showed improved reasoning skill in various problem setups [297].

Imagine three cyber-security experts searching for vulnerabilities in a code snippet. All experts will write down their line of reasoning one thought at a time, and share it with the group. They will point out where the vulnerability is found in the code. They will discuss between them the opinions, either confirming or disproving them. They will abandon an opinion when proven wrong. They will engage in the discussion until no other bugs are found. The code snippet is:

The experimentation routine was organized as follow. We first chose one sample code from each bug category. We took the longest code at Level 0. We then applied the simple prompt for each level of each sample code. We regenerated the output 3 times, and recorded the result as successful in case the model listed the vulnerability correctly.

4.4.4 RESULTS

We now provide the results of our experimentations. We subdivide them into the research questions formulated in section 4.4.2.

Were the standard datasets included during the training phase? We prompted ChatGPT with the text shown in section 4.4.3 and received the following answer.

Yes, I'm familiar with the JULIET dataset. The JULIET dataset is a collection of C and C++ code snippets [...] It is often used for training and testing static analysis tools and vulnerability scanners [...] I can provide you with a simple example of a code snippet containing a vulnerability that you might find in the JULIET dataset. [...]

The model is clearly aware of the existence of the dataset and correctly reports its purpose. However, when asked to return an example code it answers either that it cannot return the code or that it can return a code that report a vulnerability likely found also in the dataset. We alternatively tried to ask to the model if the dataset was included during its training, but the model always replied that it is not aware of the data that was used for the training. Similar results can be obtained by asking for other public datasets (e.g., Quixbugs). We can then conclude that the datasets were likely included during the training, but we need additional supporting proofs.

Were the script properly randomized before experiments? We examined the work of four major assessment studies [311, 312, 296, 313]. No study has effectively randomized the structure of the code. They simply used the code as-it-is without applying particular transformations. This greatly exposes the studies to the risk of over-estimation of performance, because the model could simply remember the code from training. A notable exception is the work in [311]. They generate a synthetic dataset using the large language model itself. While being a smart technique, the synthetic dataset can again occur in over-estimation because it was generated by the model itself. We can then conclude that previous works did not consider proper randomization of code, thus impacting the overall value of their works.

What are the reasoning limits of ChatGPT? We performed the experiments on Juliet as reported in section 4.4.3. We found that the performances dropped with the increasing level of refactoring (53.7% at Level 0, 40.7% at Level 1, 34.2% at Level 2). These results inevitably show the limits of current LLM implementations. Ideally, ChatGPT

would retain the performances at different levels, thus showing optimal generalization skills. Instead, the decline in detection provides another indication that the original dataset was included during the training.

From a qualitative perspective, ChatGPT is in general good with simple checks (e.g., **CWE259** Hardcoded password, **CWE615** Info Exposure by comment). It also shows remarkable performances on more sophisticated checks (e.g., **CWE415** Double Free, 674 Uncontrolled Recursion). It is however bad in other, seemingly simple examples (e.g., **CWE758** Undefined Behaviour, **CWE478** Missing Default Case in Switch). What emerges clearly is eventually its inadequacy to detect more complex examples where layered thinking is required. For example, it cannot handle complex deferencing of pointers (**CWE128**); it does not flag missing check on lock unlocking (**CWE667**); it does not realize that the code is changing password without authentication (**CWE620**). These examples clearly exemplify the limits of the current ChatGPT model, that cannot be seen as a practical substitute of traditional vulnerability detectors. It can however serve as an initial check to spot simple errors.

As for the use of more sophisticated prompts, we haven't experienced much change in the results. However, we have noticed a general improvement in the response, which actually becomes more like a flow of human reasoning. In particular, here are two examples in which the use of Tree of Thought actually allowed to identify the problem where the simple prompt had failed. In the first case (**CWE196** - Unsigned to Signed Conversion Error) the normal prompt suggests that the difference of two numbers can lead to a negative value, which in turn can determine stack overflow when used to access an array.

Stack Buffer Overflow: Declaring a variable-length array 'buf' with a size determined by the value of 'len' can lead to a stack buffer overflow if 'len' is negative or too large, potentially allowing for arbitrary code execution or crashes.

The complex prompt however is more specific and identifies that the value is indeed negative.

Expert 1: Alright, let's begin reviewing this code snippet. My first observation is that we're defining a char array buf with a size of len. len is calculated as a - b, which is 5 - 6, resulting in -1. This could potentially lead to an issue with the size of the buf array.

The second example (**CWE176** - Improper Handling of Unicode Encoding) regards a buffer overflow due to lack of boundaries checking. The normal prompt spots the issues at Level 0, while missing the detection at Level 1. Instead, the complex prompt continues to work.

Expert 1: Agreed. So, we've identified the first vulnerability - the potential buffer overflow in the dataBuffer. Let's move on to the next potential issue. In the second conditional block, we have calls to WideCharToMultiByte to convert rfukuwx to a multi-byte string. However, the size of the qclbftme buffer is hardcoded as 10 characters. This could lead to buffer overflow if the converted string is longer than that.

We can then conclude that using more sophisticated prompting techniques have a positive effect on the text output by the model.

4.4.5 DISCUSSION

Recent advances in the field of generative **AI** methodologies have exerted considerable influence in the domain of software engineering. **AI**-driven strategies have played a central role in addressing common challenges faced by developers, including correcting faulty programs. A notable advance in this area is the increasing adoption of **LLMs**, such as the OpenAI model called Codex. Among these models, ChatGPT, a prominent **LLM**, attracted considerable attention. It works as a versatile tool for engaging in source-focused dialogues, offering recommendations

for changes, providing explanations and even creating snippets of code. Typically, these demonstrations revolve around established benchmarks, which may have played a role in model formation, thus raising concerns about potential data loss. To assess the feasibility of using an **LLM** as an efficient programming assistant, it becomes essential to examine its capabilities in the context of the unresolved challenges.

In this article, we present an empirical investigation of ChatGPT's potential as an automated programming assistant, with a specific emphasis on its performance in vulnerability detection activities. The study evaluates the effectiveness of ChatGPT through the use of the widely recognized Juliet dataset, which serves as a benchmark to assess its ability to detect vulnerabilities. To achieve this, we performed three levels of code refactoring and applied two different stress methodologies. The results indicate a decrease in performance, from an initial value of 54% to 34%. This decrease raises the possibility of the inclusion of the dataset during the process of formation of the model. Furthermore, the integration of a more advanced stress technique seems to produce more precise responses, albeit with some limitations. Our findings offer valuable insights into the evolution of **LLMs** in programming assistance, with a particular emphasis on the importance of stress engineering, providing a deeper understanding of the practical utility of ChatGPT in the field of software engineering.

4.5 CONCLUSION

In this dissertation, we delved into the field of **IOT** identifications. Chapter 2 focused on identities based on a single attribute, acting as the most direct illustration of identity. The predominant technology in this field involves cryptographic secrets (Section 2.1), both in the form of symmetrical and asymmetric keys. Several challenges stem from this approach, including the cost of storing sensitive data in low-level devices and the operational interdependence of cryptographic algorithms. To address these challenges, **PUFs** has been suggested as a solution to conserve cryptographic secrets economically and safely (Section 2.2). Since a **PUF** acts as a blurred source, there is a need for a practical and safe fuzzy extractor to extract the precise key from the circuit. The **PUF** does offer a valuable solution. However, the growing need for complex identifications has necessitated more expressive identification methods. Chapter 3 introduced the concept of identities that include multiple attributes. These identities can, for example, amalgamate traditional password-based solutions within **RBA**, such as inferring a user's geographic location based on implicit attributes (Section 3.1.3). This poses the challenge of authenticating different identification attributes within a single token, allowing the user to select which subset of attributes to reveal during authentication. Selective dissemination techniques (Section 3.2.5) present a number of solutions to address this problem. Although multi-attribute identities offer valuable resolutions to represent the complexities of the digital sphere, these solutions address the persistent challenge of representing knowledge. The hierarchical structure required by multi-attribute identities, composed of concepts involving multiple attributes, generates a series of problems encountered during the evolution of the semantic web (Section 4.2). To address this complexity, a potential remedy could emerge from recent advances in the **LLM** field (Section 4.3). However, as demonstrated in Section 4.4, their representative capabilities remain limited and require further evaluation before practical application.

In general, the **IDOT** domain represents a dynamic field that blends technologies from different research fields. Currently, it is a trending and engaging topic, witnessing the emergence of innovative and advanced solutions aimed at addressing the multiplicity of challenges within it. However, further research remains essential to prop-

erly strengthen identities in the field of **IOT**. Our aspiration is that this thesis can serve as a cornerstone, inspiring the introduction of pioneering concepts and promoting new directions of research.

References

- [1] S. Cresci, R. Di Pietro, M. Petrocchi, A. Spognardi, and M. Tesconi, “Fame for sale: Efficient detection of fake twitter followers,” *Decision Support Systems*, vol. 80, pp. 56–71, 2015. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167923615001803>
- [2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis *et al.*, “Understanding the mirai botnet,” in *26th USENIX security symposium (USENIX Security 17)*, 2017, pp. 1093–1110.
- [3] Statista, “Annual number of ransomware attempts worldwide from 2017 to 2022,” 2023. [Online]. Available: <https://www.statista.com/statistics/494947/ransomware-attempts-per-year-worldwide/>
- [4] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, “Survey on blockchain for internet of things,” *Computer Communications*, vol. 136, pp. 10–29, 2019.
- [5] A. Čolaković and M. Hadžialić, “Internet of things (iot): A review of enabling technologies, challenges, and open research issues,” *Computer networks*, vol. 144, pp. 17–39, 2018.
- [6] A. Whitmore, A. Agarwal, and L. Da Xu, “The internet of things—a survey of topics and trends,” *Information systems frontiers*, vol. 17, pp. 261–274, 2015.
- [7] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of things: A survey on enabling technologies, protocols, and applications,” *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.
- [8] Statista, “Number of internet of things (iot) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030,” <https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/>, accessed: 2023-09-23.
- [9] S. Krčo, B. Pokrić, and F. Carrez, “Designing iot architecture (s): A european perspective,” in *2014 IEEE world forum on internet of things (WF-IoT)*. IEEE, 2014, pp. 79–84.
- [10] O. Novo, “Blockchain meets iot: An architecture for scalable access management in iot,” *IEEE internet of things journal*, vol. 5, no. 2, pp. 1184–1195, 2018.
- [11] H. Guo, J. Ren, D. Zhang, Y. Zhang, and J. Hu, “A scalable and manageable iot architecture based on transparent computing,” *Journal of Parallel and Distributed Computing*, vol. 118, pp. 5–13, 2018.
- [12] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
- [13] R. Chen, J. Guo, and F. Bao, “Trust management for soa-based iot and its application to service composition,” *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 482–495, 2014.

- [14] P. Sethi, S. R. Sarangi *et al.*, “Internet of things: architectures, protocols, and applications,” *Journal of electrical and computer engineering*, vol. 2017, 2017.
- [15] A. Jøsang, J. Fabre, B. Hay, J. Dalziel, and S. Pope, “Trust requirements in identity management,” in *Proceedings of the 2005 Australasian workshop on Grid computing and e-research-Volume 44*, 2005, pp. 99–108.
- [16] J. Camenisch, A. Shelat, D. Sommer, S. Fischer-Hübner, M. Hansen, H. Krasemann, G. Lacoste, R. Leenes, and J. Tseng, “Privacy and identity management for everyone,” in *Proceedings of the 2005 workshop on Digital identity management*, 2005, pp. 20–27.
- [17] S. Clauß and M. Köhntopp, “Identity management and its support of multilateral security,” *Computer Networks*, vol. 37, no. 2, pp. 205–219, 2001.
- [18] J. M. Such, A. Espinosa, A. Garcia-Fornes, and V. Botti, “Partial identities as a foundation for trust and reputation,” *Engineering Applications of Artificial Intelligence*, vol. 24, no. 7, pp. 1128–1136, 2011.
- [19] M. Dabrowski and P. Pacyna, “Generic and complete three-level identity management model,” in *2008 Second international conference on emerging security information, systems and technologies*. IEEE, 2008, pp. 232–237.
- [20] A. F. Gomez-Skarmeta, P. Martinez-Julia, J. Girao, and A. Sarma, “Identity based architecture for secure communication in future internet,” in *Proceedings of the 6th ACM workshop on Digital identity management*, 2010, pp. 45–48.
- [21] G. Alpár, L. Batina, L. Batten, V. Moonsamy, A. Krasnova, A. Guellier, and I. Natgunanathan, “New directions in iot privacy using attribute-based authentication,” in *Proceedings of the ACM international conference on computing frontiers*, 2016, pp. 461–466.
- [22] P. Bichsel, J. Camenisch, M. Dubovitskaya, R. R. Enderlein, S. Krenn, I. Krontiris, A. Lehmann, G. Neven, C. Paquin, F.-S. Preiss *et al.*, “An architecture for privacy-abcs,” *Attribute-based Credentials for Trust: Identity in the Information Society*, pp. 11–78, 2015.
- [23] A. Jøsang and S. Pope, “User centric identity management,” in *AusCERT Asia Pacific information technology security conference*, vol. 22. Citeseer, 2005, p. 2005.
- [24] J. Hughes and E. Maler, “Security assertion markup language (saml) v2. 0 technical overview,” *OASIS SSTC Working Draft sstc-saml-tech-overview-2.0-draft-08*, vol. 13, p. 12, 2005.
- [25] S. Cantor and T. Scavo, “Shibboleth architecture,” *Protocols and Profiles*, vol. 10, no. 16, p. 29, 2005.
- [26] S. Li, L. D. Xu, and S. Zhao, “The internet of things: a survey,” *Information systems frontiers*, vol. 17, pp. 243–259, 2015.
- [27] E. Vasilomanolakis, J. Daubert, M. Luthra, V. Gazis, A. Wiesmaier, and P. Kikiras, “On the security and privacy of internet of things architectures and systems,” in *2015 international workshop on secure internet of things (SIoT)*. IEEE, 2015, pp. 49–57.
- [28] D. Van Thanh and I. Jorstad, “The ambiguity of identity,” *Teletronikk*, vol. 103, no. 3/4, p. 3, 2007.
- [29] P. Forrest, “The identity of indiscernibles,” 1996.

- [30] C. L. Abad and R. I. Bonilla, "An analysis on the schemes for detecting and preventing arp cache poisoning attacks," in *International Conference on Distributed Computing Systems Workshops (ICDCSW '07)*, 2007, pp. 60–60.
- [31] A. Herzberg and H. Shulman, "Antidotes for dns poisoning by off-path adversaries," in *International Conference on Availability, Reliability and Security (ARES '12)*, 2012, pp. 262–267.
- [32] D. D. Dinu and M. Togan, "Dhcp server authentication using digital certificates," in *International Conference on Communications (COMM '14)*, 2014, pp. 1–6.
- [33] T. Ehrenkranz and J. Li, "On the state of ip spoofing defense," *ACM Transactions on Internet Technology (TOIT)*, vol. 9, no. 2, May 2009. [Online]. Available: <https://doi.org/10.1145/1516539.1516541>
- [34] K. M. J. Haataja and K. Hypponen, "Man-in-the-middle attacks on bluetooth: a comparative analysis, a novel attack, and countermeasures," in *International Symposium on Communications, Control and Signal Processing (ISCCSP '08)*, 2008, pp. 1096–1102.
- [35] S. Akter, S. Chellappan, T. Chakraborty, T. A. Khan, A. Rahman, and A. B. M. Alim Al Islam, "Man-in-the-middle attack on contactless payment over nfc communications: Design, implementation, experiments and detection," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.
- [36] D. Bradbury, "Hacking wifi the easy way," *Network Security*, vol. 2011, no. 2, pp. 9 – 12, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1353485811700149>
- [37] K. L. Chiew, K. S. C. Yong, and C. L. Tan, "A survey of phishing attacks: their types, vectors and technical approaches," *Expert Systems with Applications*, vol. 106, pp. 1 – 20, 2018. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0957417418302070>
- [38] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A survey on scada systems: Secure protocols, incidents, threats and tactics," *IEEE Communications Surveys Tutorials*, vol. 22, no. 3, pp. 1942–1976, 2020.
- [39] L. Franceschi-Bicchierai, "Hackers could break into your monitor to spy on you and manipulate your pixels," 2016. [Online]. Available: <https://www.vice.com/en/article/jpgdzb/hackers-could-break-into-your-monitor-to-spy-on-you-and-manipulate-your-pixels>
- [40] A. Cui and J. Kataria, "A monitor darkly: Reversing and exploiting ubiquitous osd controllers - def con 24," 2016. [Online]. Available: https://www.youtube.com/watch?v=zvP2FEfOSsk&feature=youtu.be&t=1619&ab_channel=DEFCONConference
- [41] X. Roche and other contributors, "Httrack website copier," 2020. [Online]. Available: <https://www.httrack.com/>
- [42] ooowebhost.com, "ooowebhost homepage," 2020. [Online]. Available: <https://www.000webhost.com/>
- [43] Auvideo, "B101 hdmi to csi-2 bridge (15 pin fpc)," 2016. [Online]. Available: <https://auvideo.eu/b101-hdmi-to-csi-2-bridge-15-pin-fpc/>
- [44] L. kernel, "Video for linux api," 2015. [Online]. Available: <https://www.kernel.org/doc/html/v4.9/media/uapi/v4l/v4l2.html>

- [45] R. P. foundation, “Camera module v2,” 2016. [Online]. Available: <https://www.raspberrypi.org/products/camera-module-v2/?resellerType=home>
- [46] P. S. Foundation, “Python 3.8 docs,” 2020. [Online]. Available: <https://docs.python.org/3.8/contents.html>
- [47] OpenCV, “Opencv homepage,” 2020. [Online]. Available: <https://opencv.org/>
- [48] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “Orb: An efficient alternative to sift or surf,” in *International Conference on Computer Vision (ICCV ’11)*, 2011, pp. 2564–2571.
- [49] OpenCV, “Opencv: Template matching,” 2020. [Online]. Available: https://docs.opencv.org/master/d4/dc6/tutorial_py_template_matching.html
- [50] B. Tag, J. Shimizu, C. Zhang, K. Kunze, N. Ohta, and K. Sugiura, “In the eye of the beholder: The impact of frame rate on human eye blink,” in *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*, ser. CHI EA ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 2321–2327. [Online]. Available: <https://doi.org/10.1145/2851581.2892449>
- [51] M. Conti, N. Dragoni, and V. Lesyk, “A survey of man in the middle attacks,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016.
- [52] A. Ornaghi and M. Valleri, “Man in the middle attacks,” 2003. [Online]. Available: <http://www.blackhat.com/presentations/bh-europe-03/bh-europe-03-valleri.pdf>
- [53] N. Karapanos and S. Capkun, “On the effective prevention of tls man-in-the-middle attacks in web applications,” in *USENIX Security Symposium*, ser. SEC’14. USA: USENIX Association, 2014, p. 671–686.
- [54] I. Dacosta, M. Ahamad, and P. Traynor, “Trust no one else: Detecting mitm attacks against ssl/tls without third-parties,” in *European Symposium on Research in Computer Security (ESORICS ’12)*, S. Foresti, M. Yung, and F. Martinelli, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 199–216.
- [55] G. Huston, M. Rossi, and G. Armitage, “Securing bgp — a literature survey,” *IEEE Communications Surveys Tutorials*, vol. 13, no. 2, pp. 199–222, 2011.
- [56] Y. Song, X. Hu, and Z. Lan, “The gsm/umts phone number catcher,” in *International Conference on Multimedia Information Networking and Security (MINES ’11)*, 2011, pp. 520–523.
- [57] Z. Ahmadian, S. Salimi, and A. Salahi, “New attacks on umts network access,” in *Wireless Telecommunications Symposium (WTS 2009)*, 2009, pp. 1–6.
- [58] D.-Z. Sun, Y. Mu, and W. Susilo, “Man-in-the-middle attacks on secure simple pairing in bluetooth standard v5.0 and its countermeasure,” *Personal Ubiquitous Computing*, vol. 22, no. 1, p. 55–67, Feb. 2018. [Online]. Available: <https://doi.org/10.1007/s00779-017-1081-6>
- [59] K. Hypponen and K. M. J. Haataja, ““nino” man-in-the-middle attack on bluetooth secure simple pairing,” in *IEEE/IFIP International Conference in Central Asia on Internet (ICI ’07)*, 2007, pp. 1–5.
- [60] B. Bhushan, G. Sahoo, and A. K. Rai, “Man-in-the-middle attack in wireless and computer networking — a review,” in *International Conference on Advances in Computing, Communication Automation (ICACCA ’17) (Fall)*, 2017, pp. 1–6.

- [61] U. Meyer and S. Wetzel, "A man-in-the-middle attack on umts," in *ACM Workshop on Wireless Security*, ser. WiSe '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 90–97. [Online]. Available: <https://doi.org/10.1145/1023646.1023662>
- [62] L. Zhang, W. Jia, S. Wen, and D. Yao, "A man-in-the-middle attack on 3g-wlan interworking," in *International Conference on Communications and Mobile Computing (CMC '10)*, vol. 1, 2010, pp. 121–125.
- [63] F. Ahmad, A. Adnane, V. Franqueira, F. Kurugollu, and L. Liu, "Man-in-the-middle attacks in vehicular ad-hoc networks: Evaluating the impact of attackers' strategies," *Sensors*, vol. 18, no. 11, p. 4040, Nov 2018. [Online]. Available: <http://dx.doi.org/10.3390/s18114040>
- [64] F. Lemarchand, C. Marlin, F. Montreuil, E. Nogues, and M. Pelcat, "Electro-magnetic side-channel attack through learned denoising and classification," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 2882–2886.
- [65] N. Zhang, Y. Lu, Q. Cui, and Y. Wang, "Investigation of unintentional video emanations from a vga connector in the desktop computers," *IEEE Transactions on Electromagnetic Compatibility*, vol. 59, no. 6, pp. 1826–1834, 2017.
- [66] F. Elibol, U. Sarac, and I. Erer, "Realistic eavesdropping attacks on computer displays with low-cost and mobile receiver system," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, 2012, pp. 1767–1771.
- [67] A. Davis, "What the heck? security implications of hdmi ethernet channel and other related protocols," 2013. [Online]. Available: <https://www.nccgroup.com/uk/our-research/hdmi-ethernet-channel/>
- [68] J. Smith, "High-def fuzzing : Exploring vulnerabilities in hdmiccc," 2015. [Online]. Available: <https://media.defcon.org/DEF%20CON%2023/DEF%20CON%2023%20presentations/DEF%20CON%2023%20-%20Joshua-Smith-High-Def-Fuzzing-Exploitation-Over-HDMI-CEC-UPDATED.pdf>
- [69] L. P. Rondon, L. Babun, K. Akkaya, and A. S. Uluagac, "Hdmi-walk: Attacking hdmi distribution networks via consumer electronic control protocol," in *Annual Computer Security Applications Conference*, ser. ACSAC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 650–659. [Online]. Available: <https://doi.org/10.1145/3359789.3359841>
- [70] D. C. P. LLC, "Hdcp specifications," 2020. [Online]. Available: <https://www.digital-cp.com/hdcp-specifications>
- [71] B. A. Forouzan and D. Mukhopadhyay, *Cryptography and network security*. Mc Graw Hill Education (India) Private Limited New York, NY, USA:, 2015, vol. 12.
- [72] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.
- [73] S. Jeong, J. Hwang, H. Kwon, and D. Shin, "A cfi countermeasure against got overwrite attacks," *IEEE Access*, vol. 8, pp. 36 267–36 280, 2020.
- [74] S. A. Islam and S. Katkoori, "Safecontroller: Efficient and transparent control-flow integrity for rtl design," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2020, pp. 270–275.

- [75] H. Xia, “Capability memory protection for embedded systems,” Ph.D. dissertation, University of Cambridge, 2020.
- [76] M. Conti, R. Di Pietro, A. Gabrielli, L. V. Mancini, and A. Mei, “The smallville effect: social ties make mobile networks more secure against node capture attack,” in *Proceedings of the 8th ACM international workshop on Mobility management and wireless access*, 2010, pp. 99–106.
- [77] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei, “Emergent properties: detection of the node-capture attack in mobile wireless sensor networks,” in *Proceedings of the first ACM conference on Wireless network security*, 2008, pp. 214–219.
- [78] A. Ibrahim, A.-R. Sadeghi, G. Tsudik, and S. Zeitouni, “Darpa: Device attestation resilient to physical attacks,” in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, ser. WiSec '16. New York, NY, USA: ACM, 2016, pp. 171–182. [Online]. Available: <http://doi.acm.org/10.1145/2939918.2939938>
- [79] A. Ibrahim, A.-R. Sadeghi, and S. Zeitouni, “Seed: secure non-interactive attestation for embedded devices,” in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2017, pp. 64–74.
- [80] F. Kohnhäuser, N. Büscher, and S. Katzenbeisser, “A practical attestation protocol for autonomous embedded systems,” in *4th IEEE European Symposium on Security and Privacy (EuroSec'19)*, June 2019. [Online]. Available: <http://tubiblio.ulb.tu-darmstadt.de/114633/>
- [81] A. Ibrahim, A.-R. Sadeghi, and G. Tsudik, “Us-aid: Unattended scalable attestation of iot devices,” in *37th IEEE International Symposium on Reliable Distributed Systems*, July 2018. [Online]. Available: <https://ieeexplore.ieee.org/document/8613950>
- [82] B. Gong, Y. Zhang, and Y. Wang, “A remote attestation mechanism for the sensing layer nodes of the internet of things,” *Future Generation Computer Systems*, vol. 78, pp. 867–886, 2018.
- [83] A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik, “A minimalist approach to remote attestation,” in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–6.
- [84] I. Cisco Systems. Why ip is the right foundation for the smart grid. Last access November 2020. [Online]. Available: <https://www.cisco.com/c/dam/assets/docs/c11-581079-wp.pdf>
- [85] K. Mandula, R. Parupalli, C. A. Murty, E. Magesh, and R. Lunagariya, “Mobile based home automation using internet of things (iot),” in *2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. IEEE, 2015, pp. 340–343.
- [86] Z. Shelby and C. Bormann, *6LoWPAN: The wireless embedded Internet*. John Wiley & Sons, 2011, vol. 43.
- [87] T. Group. Thread. Last access November 2020. [Online]. Available: <https://www.threadgroup.org/>
- [88] M. Kylänpää and A. Rantala, “Remote attestation for embedded systems,” in *Security of Industrial Control Systems and Cyber Physical Systems*, A. Bécue, N. Cuppens-Boulahia, F. Cuppens, S. Katsikas, and C. Lambrinoudakis, Eds. Cham: Springer International Publishing, 2016, pp. 79–92.

- [89] A. Varga, “Omnet++,” in *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59.
- [90] K. Eldefrawy and G. Tsudik, “Advancing remote attestation via computer-aided formal verification of designs and synthesis of executables: Opinion,” in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec ’19. New York, NY, USA: ACM, 2019, pp. 45–48. [Online]. Available: <http://doi.acm.org/10.1145/3317549.3323403>
- [91] T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik, “C-flat: Control-flow attestation for embedded systems software,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’16. New York, NY, USA: ACM, 2016, pp. 743–754. [Online]. Available: <http://doi.acm.org/10.1145/2976749.2978358>
- [92] T. Abera, R. Bahmani, F. Brasser, A. Ibrahim, A. Sadeghi, and M. Schunter, “Diat: Data integrity attestation for resilient collaboration of autonomous systems,” in *26th Annual Network & Distributed System Security Symposium (NDSS)*. The Internet Society, January 2019. [Online]. Available: <http://tubiblio.ulb.tu-darmstadt.de/110632/>
- [93] S. Schulz, A. Schaller, F. Kohnhäuser, and S. Katzenbeisser, “Boot attestation: Secure remote reporting with off-the-shelf iot sensors,” in *Computer Security – ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Sneekenes, Eds. Cham: Springer International Publishing, 2017, pp. 437–455.
- [94] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 4, pp. 149–160, Aug. 2001. [Online]. Available: <http://doi.acm.org/10.1145/964723.383071>
- [95] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup protocol for internet applications,” *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003. [Online]. Available: <http://dx.doi.org/10.1109/TNET.2002.808407>
- [96] P. Zave, “How to make chord correct (using a stable base),” *CoRR*, vol. abs/1502.06461, 2015. [Online]. Available: <http://arxiv.org/abs/1502.06461>
- [97] W. Diffie and M. Hellman, “New directions in cryptography,” *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.
- [98] A. Bhatt and J. Patoliya, “Cost effective digitization of home appliances for home automation with low-power wifi devices,” in *2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, 2016, pp. 643–648.
- [99] A. Rayes and S. Salam, “The internet in iot,” in *Internet of Things From Hype to Reality*. Springer, 2019, pp. 37–65.
- [100] F. Kohnhäuser, N. Büscher, S. Gabmeyer, and S. Katzenbeisser, “Scapi: A scalable attestation protocol to detect software and physical attacks,” in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec ’17. New York, NY, USA: ACM, 2017, pp. 75–86. [Online]. Available: <http://doi.acm.org/10.1145/3098243.3098255>

- [101] D. Dolev and A. C. Yao, “On the security of public key protocols,” in *Proceedings of the 22Nd Annual Symposium on Foundations of Computer Science*, ser. SFCS ’81. Washington, DC, USA: IEEE Computer Society, 1981, pp. 350–357. [Online]. Available: <https://doi.org/10.1109/SFCS.1981.32>
- [102] M. Ambrosin, M. Conti, A. Ibrahim, G. Neven, A.-R. Sadeghi, and M. Schunter, “Sana: Secure and scalable aggregate network attestation,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’16. New York, NY, USA: ACM, 2016, pp. 731–742. [Online]. Available: <http://doi.acm.org/10.1145/2976749.2978335>
- [103] F. Toffalini, E. Losiouk, A. Biondo, J. Zhou, and M. Conti, “Scarr: Scalable runtime remote attestation for complex systems,” in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. Chaoyang District, Beijing: USENIX Association, Sep. 2019, pp. 121–134. [Online]. Available: <https://www.usenix.org/conference/raid2019/presentation/toffalini>
- [104] G. Dessouky, S. Zeitouni, T. Nyman, A. Paverd, L. Davi, P. Koeberl, N. Asokan, and A.-R. Sadeghi, “Lo-fat: Low-overhead control flow attestation in hardware,” in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.
- [105] D. Challener, “Trusted platform module,” *Encyclopedia of Cryptography and Security*, pp. 1308–1310, 2011.
- [106] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, A. Ibrahim, Y. Jin, and A.-R. Sadeghi, “Atrium: Runtime attestation resilient under memory attacks,” in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 384–391.
- [107] F. Kohnhäuser, N. Büscher, and S. Katzenbeisser, “Salad: Secure and lightweight attestation of highly dynamic and disruptive networks,” in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 329–342. [Online]. Available: <https://doi.org/10.1145/3196494.3196544>
- [108] M. Liberatore and B. N. Levine, “Inferring the source of encrypted http connections,” in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS ’06. New York, NY, USA: Association for Computing Machinery, 2006, p. 255–263. [Online]. Available: <https://doi.org/10.1145/1180405.1180437>
- [109] M. Conti, G. Rigoni, and F. Toffalini, “Asaint: a spy app identification system based on network traffic,” in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, 2020, pp. 1–8.
- [110] M. Abe, “Mix-networks on permutation networks,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 1999, pp. 258–273.
- [111] H. Feistel, “Cryptography and computer privacy,” *Scientific american*, vol. 228, no. 5, pp. 15–23, 1973.
- [112] R. Pi. Raspberry pi zero. [Online]. Available: <https://www.raspberrypi.org/products/raspberry-pi-zero/>
- [113] —, “Raspberry pi 3 model b,” *online*. (<https://www.raspberrypi.org>), 2015.
- [114] J. Salowey, A. Choudhury, and D. McGrew, “Aes galois counter mode (gcm) cipher suites for tls,” *Request for Comments*, vol. 5288, 2008.

- [115] J. Winter, “Trusted computing building blocks for embedded linux-based arm trustzone platforms,” in *Proceedings of the 3rd ACM Workshop on Scalable Trusted Computing*, ser. STC ’08. New York, NY, USA: ACM, 2008, pp. 21–30. [Online]. Available: <http://doi.acm.org/10.1145/1456455.1456460>
- [116] Linaro. (2015) Op-tee. Last access June 2019. [Online]. Available: https://github.com/OP-TEE/optee_os
- [117] N. Asokan, F. Brasser, A. Ibrahim, A.-R. Sadeghi, M. Schunter, G. Tsudik, and C. Wachsmann, “Seda: Scalable embedded device attestation,” in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’15. New York, NY, USA: ACM, 2015, pp. 964–975. [Online]. Available: <http://doi.acm.org/10.1145/2810103.2813670>
- [118] X. Carpent, K. ElDefrawy, N. Rattanaivanon, and G. Tsudik, “Lightweight swarm attestation: A tale of two lisa-s,” in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS ’17. New York, NY, USA: ACM, 2017, pp. 86–100. [Online]. Available: <http://doi.acm.org/10.1145/3052973.3053010>
- [119] M. Ambrosin, M. Conti, R. Lazzeretti, M. M. Rabbani, and S. Ranise, “Pads: Practical attestation for highly dynamic swarm topologies,” in *2018 International Workshop on Secure Internet of Things (SIoT)*, 2018, pp. 18–27.
- [120] A. Broder and M. Mitzenmacher, “Network applications of bloom filters: A survey,” *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004. [Online]. Available: <https://doi.org/10.1080/15427951.2004.10129096>
- [121] M. Hiller, “Key derivation with physical unclonable functions,” Ph.D. dissertation, Technische Universität München, 2016.
- [122] M. Roel, “Physically unclonable functions: Constructions, properties and applications,” *Katholieke Universiteit Leuven, Belgium*, pp. 148–160, 2012.
- [123] J. Obermaier, V. Immler, M. Hiller, and G. Sigl, “A measurement system for capacitive puf-based security enclosures,” in *Proceedings of the 55th Annual Design Automation Conference*, 2018, pp. 1–6.
- [124] C.-H. Chang, Y. Zheng, and L. Zhang, “A retrospective and a look forward: Fifteen years of physical unclonable function advancement,” *IEEE Circuits and Systems Magazine*, vol. 17, no. 3, pp. 32–62, 2017.
- [125] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, “Fuzzy extractors: How to generate strong keys from biometrics and other noisy data,” *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.
- [126] X. Boyen, “Reusable cryptographic fuzzy extractors,” in *Proceedings of the 11th ACM Conference on Computer and Communications Security*. Association for Computing Machinery, 2004, p. 82–91.
- [127] Y. Wen and S. Liu, “Reusable fuzzy extractor from lwe,” in *Information Security and Privacy: 23rd Australasian Conference, ACISP 2018, Wollongong, NSW, Australia, July 11-13, 2018, Proceedings*, 2018, pp. 13–27.

- [128] —, “Robustly reusable fuzzy extractor from standard assumptions,” in *Advances in Cryptology—ASIACRYPT 2018: 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2–6, 2018, Proceedings, Part III* 24. Springer, 2018, pp. 459–489.
- [129] B. Fuller, X. Meng, and L. Reyzin, “Computational fuzzy extractors,” *Information and Computation*, vol. 275, p. 104602, 2020.
- [130] R. Canetti, B. Fuller, O. Paneth, L. Reyzin, and A. Smith, “Reusable fuzzy extractors for low-entropy distributions,” *Journal of Cryptology*, vol. 34, pp. 1–33, 2021.
- [131] J. Woo, J. Kim, and J. H. Park, “Robust and reusable fuzzy extractors from non-uniform learning with errors problem,” *Computers, Materials & Continua*, vol. 74, no. 1, 2023.
- [132] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, and A. Smith, “Secure remote authentication using biometric data,” in *24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005. Proceedings* 24, 2005, pp. 147–163.
- [133] D. Apon, C. Cho, K. Eldefrawy, and J. Katz, “Efficient, reusable fuzzy extractors from lwe,” in *Cyber Security Cryptography and Machine Learning: First International Conference, CSCML 2017, Beer-Sheva, Israel, June 29–30, 2017, Proceedings* 1. Springer, 2017, pp. 1–18.
- [134] B. Chen, T. Ignatenko, F. Willems, R. Maes, E. van der Sluis, and G. Selimis, “High-rate error correction schemes for sram-pufs based on polar codes,” *arXiv preprint arXiv:1701.07320*, 2017.
- [135] M. Seo, S. Kim, D. H. Lee, and J. H. Park, “Emblem:(r) lwe-based key encapsulation with a new multi-bit encoding method,” *International Journal of Information Security*, vol. 19, pp. 383–399, 2020.
- [136] R. Maes, V. Van Der Leest, E. Van Der Sluis, and F. Willems, “Secure key generation from biased pufs,” in *Cryptographic Hardware and Embedded Systems—CHES 2015: 17th International Workshop, Saint-Malo, France, September 13–16, 2015, Proceedings* 17. Springer, 2015, pp. 517–534.
- [137] F. Wilde, B. M. Gammel, and M. Pehl, “Spatial correlation analysis on physical unclonable functions,” *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 6, pp. 1468–1480, 2018.
- [138] M. T. Rahman, A. Hosey, Z. Guo, J. Carroll, D. Forte, and M. Tehranipoor, “Systematic correlation and cell neighborhood analysis of sram puf for robust and unique key generation,” *Journal of Hardware and Systems Security*, vol. 1, pp. 137–155, 2017.
- [139] I. R. Shafarevich and A. O. Remizov, *Linear algebra and geometry*. Springer Science & Business Media, 2012.
- [140] M. Liu, C. Zhou, Q. Tang, K. K. Parhi, and C. H. Kim, “A data remanence based approach to generate 100% stable keys from an sram physical unclonable function,” in *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2017, pp. 1–6.
- [141] S. Wiefeling, L. Lo Iacono, and M. Dürmuth, “Is this really you? an empirical study on risk-based authentication applied in the wild,” in *ICT Systems Security and Privacy Protection*, G. Dhillon, F. Karlsson, K. Hedström, and A. Zúquete, Eds. Cham: Springer International Publishing, 2019, pp. 134–148.

- [142] L. Diamond and M. F. Plattner, *Liberation technology: Social media and the struggle for democracy*. JHU Press, 2012.
- [143] S. Moghanizadeh, "The role of social media in iran's green movement," Master's thesis, University of Gothenburg, 2013.
- [144] P. N. Howard and M. M. Hussain, "The upheavals in egypt and tunisia: The role of digital media," *Journal of democracy*, vol. 22, no. 3, pp. 35–48, 2011.
- [145] J. H. Parmelee and S. L. Bichard, *Politics and the Twitter revolution: How tweets influence the relationship between political leaders and the public*. Lexington books, 2011.
- [146] E. Ferrara, "Disinformation and social bot operations in the run up to the 2017 french presidential election," *First Monday*, jul 2017. [Online]. Available: <https://doi.org/10.5210/2Ffm.v22i8.8005>
- [147] A. Bessi and E. Ferrara, "Social bots distort the 2016 us presidential election online discussion," *First monday*, vol. 21, no. 11-7, 2016.
- [148] P. T. Metaxas and E. Mustafaraj, "Social media and the elections," *Science*, vol. 338, no. 6106, pp. 472–473, 2012. [Online]. Available: <https://www.science.org/doi/abs/10.1126/science.1230456>
- [149] M. T. Bastos and D. Mercea, "The brexit botnet and user-generated hyperpartisan news," *Social science computer review*, vol. 37, no. 1, pp. 38–54, 2019.
- [150] J.-P. Verkamp and M. Gupta, "Five incidents, one theme: Twitter spam as a weapon to drown voices of protest," in *3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI 13)*. Washington, D.C.: USENIX Association, Aug. 2013. [Online]. Available: <https://www.usenix.org/conference/foci13/workshop-program/presentation/verkamp>
- [151] R. Rogers and S. Niederer, *The politics of social media manipulation*. Amsterdam University Press, 2020.
- [152] E. Alothali, N. Zaki, E. A. Mohamed, and H. Alashwal, "Detecting social bots on twitter: a literature review," in *2018 International conference on innovations in information technology (IIT)*. IEEE, 2018, pp. 175–180.
- [153] NewsWeek, "Joe Biden, Kamala Harris got a big social media boost from Indian troll farms," <https://www.newsweek.com/joe-biden-kamala-harris-got-big-social-media-boost-indian-troll-farms-1544047>, accessed: 2023-01-01.
- [154] N. Louise, "Indian troll companies reportedly created 738,595 fake twitter followers to follow Joe Biden and Kamala Harris," <https://techstartups.com/2020/11/02/indian-troll-companies-reportedly-created-738595-fake-twitter-followers-just-13-days-follow-joe-biden-kamala-harris/>, accessed: 2023-01-01.
- [155] TwitterAudit, "Twitteraudit," <https://twitteraudit.com/>, accessed: 2023-01-01.
- [156] FollowerAudit, "Followeraudit," <https://www.followeraudit.com/>, accessed: 2023-01-01.
- [157] SparkToro, "Sparktoro," <https://sparktoro.com/tools/fake-followers-audit>, accessed: 2023-01-01.

- [158] K.-C. Yang, E. Ferrara, and F. Menczer, “Botometer 101: Social bot practicum for computational social scientists,” *arXiv preprint arXiv:2201.01608*, 2022. [Online]. Available: <https://botometer.osome.iu.edu/api>
- [159] M. B. Torusdağ, M. Kutlu, and A. A. Selçuk, “Are we secure from bots? investigating vulnerabilities of botometer,” in *2020 5th International Conference on Computer Science and Engineering (UBMK)*, 2020, pp. 343–348.
- [160] B. Erşahin, Ö. Aktaş, D. Kılınç, and C. Akyol, “Twitter fake account detection,” in *2017 International Conference on Computer Science and Engineering (UBMK)*. IEEE, 2017, pp. 388–392.
- [161] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, “Detecting automation of twitter accounts: Are you a human, bot, or cyborg?” *IEEE Transactions on dependable and secure computing*, vol. 9, no. 6, pp. 811–824, 2012.
- [162] J. Castellini, V. Poggioni, and G. Sorbi, “Fake twitter followers detection by denoising autoencoder,” in *Proceedings of the International Conference on Web Intelligence*, 2017, pp. 195–202.
- [163] A. Khalil, H. Hajjdiab, and N. Al-Qirim, “Detecting fake followers in twitter: A machine learning approach,” *International Journal of Machine Learning and Computing*, vol. 7, no. 6, pp. 198–202, 2017.
- [164] G. Stringhini, G. Wang, M. Egele, C. Kruegel, G. Vigna, H. Zheng, and B. Y. Zhao, “Follow the green: growth and dynamics in twitter follower markets,” in *Proceedings of the 2013 conference on Internet measurement conference*, 2013, pp. 163–176.
- [165] A. Mehrotra, M. Sarreddy, and S. Singh, “Detection of fake twitter followers using graph centrality measures,” in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, 2016, pp. 499–504.
- [166] I. Twitter, “Twitter homepage,” <https://twitter.com/>, accessed: 2023-01-01.
- [167] Statista, “Twitter global mda 2022,” <https://www.statista.com/statistics/970920/monetizable-daily-active-twitter-users-worldwide/>, accessed: 2023-01-01.
- [168] R. King, “Announcing snowflake,” https://blog.twitter.com/engineering/en_us/a/2010/announcing-snowflake, accessed: 2023-01-01.
- [169] T. Archive, “Snowflake github repository,” <https://github.com/twitter-archive/snowflake/tree/snowflake-2010>, accessed: 2023-01-01.
- [170] I. Discord, “Discord api reference,” <https://discord.com/developers/docs/reference#snowflakes>, accessed: 2023-01-01.
- [171] “Twitter api homepage,” <https://developer.twitter.com/en/docs/twitter-api>, accessed: 2023-01-01.
- [172] “Python homepage,” <https://www.python.org/>, accessed: 2023-01-01.
- [173] “Leading countries based on number of twitter users as of january 2022,” <https://www.statista.com/statistics/242606/number-of-active-twitter-users-in-selected-countries/>, accessed: 2023-01-01.

- [174] M. Arora and V. Kansal, "Character level embedding with deep convolutional neural network for text normalization of unstructured data for twitter sentiment analysis," *Social Network Analysis and Mining*, vol. 9, pp. 1–14, 2019.
- [175] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, 2015.
- [176] "gensim 4.3.1," <https://pypi.org/project/gensim/>, accessed: 2023-03-11.
- [177] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan, "Finding a" kneedle" in a haystack: Detecting knee points in system behavior," in *2011 31st international conference on distributed computing systems workshops*. IEEE, 2011, pp. 166–171.
- [178] D. Leonhardt, "Kamala harris, front-runner (again)," <https://www.nytimes.com/2020/06/12/briefing/kamala-harris-mark-milley-trump-rally-your-friday-briefing.html>, accessed: 2023-01-01.
- [179] P. A. Grassi, M. E. Garcia, and J. L. Fenton, "Digital identity guidelines," *NIST special publication*, vol. 800, pp. 63–3, 2017. [Online]. Available: https://www.jpipdec.or.jp/eventseminar/event/u71kba0000009o56-att/nist_sp_800-63-3.pdf
- [180] K. Cameron, "The laws of identity," *Microsoft Corp*, vol. 12, pp. 8–11, 2005. [Online]. Available: <http://myinstantid.com/laws.pdf>
- [181] P. R. C. Lee Rainie, "Americans' complicated feelings about social media in an era of privacy concerns," 2018. [Online]. Available: <https://www.pewresearch.org/fact-tank/2018/03/27/americans-complicated-feelings-about-social-media-in-an-era-of-privacy-concerns/>
- [182] B. N. Tiffanie Turnbull, "Optus: How a massive data breach has exposed australia," 2022. [Online]. Available: <https://www.bbc.com/news/world-australia-63056838>
- [183] B. Alina Bizga, "500 million whatsapp mobile phone numbers are up for grabs on the dark web," 2022. [Online]. Available: <https://www.bbc.com/news/world-australia-63056838>
- [184] B. N. Chris Vallance, "Twitter in data-protection probe after '400 million' user details up for sale," 2022. [Online]. Available: <https://www.bbc.com/news/technology-64109777>
- [185] D. Loffreto, "What is "sovereign source authority"," 2012. [Online]. Available: <https://www.moxytongue.com/2012/02/what-is-sovereign-source-authority.html>
- [186] U. Nations, "Universal declaration of human rights," 1948. [Online]. Available: <https://www.un.org/en/about-us/universal-declaration-of-human-rights>
- [187] O. of the High Commissioner for Human Rights, "Convention on the rights of the child," 1989. [Online]. Available: <https://www.ohchr.org/en/instruments-mechanisms/instruments/convention-rights-child>
- [188] U. R. Agency, "Refugee crisis in europe," 2015. [Online]. Available: <https://www.unrefugees.org/emergencies/refugee-crisis-in-europe/>
- [189] D. W. Chadwick, "Federated identity management," in *Foundations of security analysis and design V*. Springer, 2009, pp. 96–120.

- [190] T. El Maliki and J.-M. Seigneur, "A survey of user-centric identity management technologies," in *The International Conference on Emerging Security Information, Systems, and Technologies (SECUREWARE 2007)*. IEEE, 2007, pp. 12–17.
- [191] C. Allen, "The path to self-sovereign identity," 2016. [Online]. Available: www.lifewithalacrity.com/2016/04/the-path-to-self-sovereign-identity.html
- [192] A. Tobin and D. Reed, "The inevitable rise of self-sovereign identity," *The Sovrin Foundation*, vol. 29, no. 2016, p. 18, 2016.
- [193] W. W. W. Consortium, "Verifiable credentials data model v1.1," 2022. [Online]. Available: <https://www.w3.org/TR/vc-data-model/>
- [194] D. Baars, "Towards self-sovereign identity using blockchain technology," Master's thesis, University of Twente, 2016.
- [195] Q. Stokkink and J. Pouwelse, "Deployment of a blockchain-based self-sovereign identity," in *2018 IEEE international conference on Internet of Things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*. IEEE, 2018, pp. 1336–1342.
- [196] M. S. Ferdous, F. Chowdhury, and M. O. Alassafi, "In search of self-sovereign identity leveraging blockchain technology," *IEEE access*, vol. 7, pp. 103 059–103 079, 2019.
- [197] W. W. W. Consortium, "Decentralized identifiers (dids) v1.0," 2022. [Online]. Available: <https://www.w3.org/TR/did-core/>
- [198] C. Brunner, U. Gellersdörfer, F. Knirsch, D. Engel, and F. Matthes, "Did and vc: Untangling decentralized identifiers and verifiable credentials for the web of trust," in *2020 the 3rd International Conference on Blockchain Technology and Applications*, 2020, pp. 61–66.
- [199] H. I. Portability and A. A. (HIPAA), "Minimum necessary requirement," 2003. [Online]. Available: <https://www.hhs.gov/hipaa/for-professionals/privacy/guidance/minimum-necessary-requirement/index.html>
- [200] E. D. P. Supervisor, "General data protection regulation (gdpr)," 2016. [Online]. Available: <https://gdpr-info.eu/>
- [201] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.
- [202] A. Stein and E. Teske, "Optimized baby step-giant step methods," *J. Ramanujan Math. Soc.*, vol. 20, no. 1, pp. 1–32, 2005.
- [203] D. Boneh, "The decision diffie-hellman problem," in *International algorithmic number theory symposium*. Springer, 1998, pp. 48–63.
- [204] N. Tanaka and T. Saito, "On the q-strong diffie-hellman problem," *Cryptology ePrint Archive*, 2010.
- [205] J. H. Silverman, *The arithmetic of elliptic curves*. Springer, 2009, vol. 106.

- [206] R. Dutta, R. Barua, and P. Sarkar, “Pairing-based cryptographic protocols: A survey,” *Cryptology ePrint Archive*, 2004.
- [207] S. Goldwasser, S. Micali, and C. Rackoff, “The knowledge complexity of interactive proof-systems,” in *Providing sound foundations for cryptography: On the work of shafi goldwasser and silvio micali*, 2019, pp. 203–225.
- [208] M. Blum, P. Feldman, and S. Micali, “Non-interactive zero-knowledge and its applications,” in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 329–349.
- [209] J. Camenisch and M. Stadler, “Efficient group signature schemes for large groups,” in *Advances in Cryptology—CRYPTO’97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17–21, 1997 Proceedings 17*. Springer, 1997, pp. 410–424.
- [210] C.-P. Schnorr, “Efficient signature generation by smart cards,” *Journal of cryptology*, vol. 4, pp. 161–174, 1991.
- [211] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Conference on the theory and application of cryptographic techniques*. Springer, 1986, pp. 186–194.
- [212] J. Camenisch and A. Lysyanskaya, “An efficient system for non-transferable anonymous credentials with optional anonymity revocation,” in *Advances in Cryptology—EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6–10, 2001 Proceedings 20*. Springer, 2001, pp. 93–118.
- [213] —, “A signature scheme with efficient protocols,” in *Security in Communication Networks: Third International Conference, SCN 2002 Amalfi, Italy, September 11–13, 2002 Revised Papers 3*. Springer, 2003, pp. 268–289.
- [214] —, “Signature schemes and anonymous credentials from bilinear maps,” in *Advances in Cryptology—CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15–19, 2004. Proceedings 24*. Springer, 2004, pp. 56–72.
- [215] H. foundation, “Anoncreds home page,” 2023. [Online]. Available: <https://www.hyperledger.org/projects/anoncreds>
- [216] IBM, “Idemix github repository,” 2023. [Online]. Available: <https://github.com/IBM/idemix>
- [217] H. project foundation, “Hyperledger indy,” 2023. [Online]. Available: <https://www.hyperledger.org/projects/hyperledger-indy>
- [218] A. Lysyanskaya, “Signature schemes and applications to cryptographic protocol design,” Ph.D. dissertation, Massachusetts Institute of Technology, 2002.
- [219] E. Brickell, J. Camenisch, and L. Chen, “Direct anonymous attestation,” in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 132–145.

- [220] J. Camenisch and J. Groth, “Group signatures: Better efficiency and new theoretical aspects,” in *Security in Communication Networks: 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers 4*. Springer, 2005, pp. 120–133.
- [221] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, “From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again,” in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 2012, pp. 326–349.
- [222] J. Groth, “Short pairing-based non-interactive zero-knowledge arguments,” in *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*. Springer, 2010, pp. 321–340.
- [223] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, “Quadratic span programs and succinct nizks without pcps,” in *Advances in Cryptology-EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*. Springer, 2013, pp. 626–645.
- [224] B. Parno, J. Howell, C. Gentry, and M. Raykova, “Pinocchio: Nearly practical verifiable computation,” *Communications of the ACM*, vol. 59, no. 2, pp. 103–112, 2016.
- [225] I. Damgård, “Towards practical public key systems secure against chosen ciphertext attacks,” in *Advances in Cryptology-CRYPTO’91: Proceedings 11*. Springer, 1992, pp. 445–456.
- [226] M. Botts, G. Percivall, C. Reed, and J. Davidson, “Ogc® sensor web enablement: Overview and high level architecture,” in *GeoSensor Networks: Second International Conference, GSN 2006, Boston, MA, USA, October 1-3, 2006, Revised Selected and Invited Papers*. Springer, 2008, pp. 175–190.
- [227] C. Prehofer, “Models at rest or modelling restful interfaces for the internet of things,” in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. IEEE, 2015, pp. 251–255.
- [228] N. Shadbolt, T. Berners-Lee, and W. Hall, “The semantic web revisited,” *IEEE intelligent systems*, vol. 21, no. 3, pp. 96–101, 2006.
- [229] I. Szilagyi and P. Wira, “Ontologies and semantic web for the internet of things-a survey,” in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2016, pp. 6949–6954.
- [230] F. Manola, E. Miller, B. McBride *et al.*, “Rdf primer,” *W3C recommendation*, vol. 10, no. 1-107, p. 6, 2004.
- [231] H. Hasemann, A. Kröller, and M. Pagel, “The wiselib tuplestore: a modular rdf database for the internet of things,” *arXiv preprint arXiv:1402.7228*, 2014.
- [232] C. Gutierrez, C. A. Hurtado, and A. Vaisman, “Introducing time into rdf,” *IEEE transactions on Knowledge and Data Engineering*, vol. 19, no. 2, pp. 207–218, 2006.
- [233] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, “COMET: Commonsense transformers for automatic knowledge graph construction,” Jun. 2019.
- [234] D. Brickley, R. V. Guha, and B. McBride, “Rdf schema 1.1,” *W3C recommendation*, vol. 25, no. 2004-2014, p. 10, 2014.

- [235] D. L. McGuinness, F. Van Harmelen *et al.*, “Owl web ontology language overview,” *W3C recommendation*, vol. 10, no. 10, p. 2004, 2004.
- [236] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, S. Rudolph *et al.*, “Owl 2 web ontology language primer,” *W3C recommendation*, vol. 27, no. 1, p. 123, 2009.
- [237] W. W. W. Consortium *et al.*, “Owl 2 web ontology language document overview,” 2012.
- [238] M. Compton, P. Barnaghi, L. Bermudez, R. Garcia-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog *et al.*, “The ssn ontology of the w3c semantic sensor network incubator group,” *Journal of Web Semantics*, vol. 17, pp. 25–32, 2012.
- [239] I. de Recherche en Informatique de Toulouse. (2016) Semantic actuator network. Last access October 2023. [Online]. Available: <https://www.irit.fr/recherches/MELODI/ontologies/SAN.owl>
- [240] F. Ganz, P. Barnaghi, and F. Carrez, “Automated semantic knowledge acquisition from sensor data,” *IEEE Systems Journal*, vol. 10, no. 3, pp. 1214–1225, 2014.
- [241] P. Warren, P. Mulholland, T. Collins, and E. Motta, “The usability of description logics: understanding the cognitive difficulties presented by description logics,” in *The Semantic Web: Trends and Challenges: 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings 11*. Springer, 2014, pp. 550–564.
- [242] M. Vigo, S. Bail, C. Jay, and R. Stevens, “Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design,” *International Journal of Human-Computer Studies*, vol. 72, no. 12, pp. 835–845, 2014.
- [243] N. A. Abdul Manaf, S. Bechhofer, and R. Stevens, “The current state of skos vocabularies on the web,” in *Extended Semantic Web Conference*. Springer, 2012, pp. 270–284.
- [244] S. Jupp, S. Bechhofer, and R. Stevens, “Skos with owl: Don’t be full-ish!” in *OWLED*, vol. 432, 2008, pp. 2009–2.
- [245] M. Horridge and S. Bechhofer, “The owl api: A java api for owl ontologies,” *Semantic web*, vol. 2, no. 1, pp. 11–21, 2011.
- [246] R. Taelman, M. Vander Sande, and R. Verborgh, “GraphQL-ld: linked data querying with graphql,” in *ISWC2018, the 17th International Semantic Web Conference*, 2018, pp. 1–4.
- [247] R. Meusel, P. Petrovski, and C. Bizer, “The webdatacommons microdata, rdfa and microformat dataset series,” in *The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I 13*. Springer, 2014, pp. 277–292.
- [248] K. Rabbani, M. Lissandrini, and K. Hose, “Shacl and shex in the wild: a community survey on validating shapes generation and adoption,” in *Companion Proceedings of the Web Conference 2022*, 2022, pp. 260–263.
- [249] M. d’Aquin and N. F. Noy, “Where to publish and find ontologies? a survey of ontology libraries,” *Journal of Web Semantics*, vol. 11, pp. 96–111, 2012.

- [250] M. Sabou and M. Fernandez, “Ontology (network) evaluation,” in *Ontology engineering in a networked world*. Springer, 2011, pp. 193–212.
- [251] M. Poveda Villalón, “Ontology evaluation: a pitfall-based approach to ontology diagnosis,” Ph.D. dissertation, ETSI_Informatica, 2016.
- [252] A. Ghazvinian, N. F. Noy, and M. A. Musen, “How orthogonal are the obo foundry ontologies?” in *Journal of biomedical semantics*, vol. 2. Springer, 2011, pp. 1–14.
- [253] M. R. Kamdar, T. Tudorache, and M. A. Musen, “A systematic analysis of term reuse and term overlap across biomedical ontologies,” *Semantic web*, vol. 8, no. 6, pp. 853–871, 2017.
- [254] H. Stuckenschmidt, C. Parent, and S. Spaccapietra, *Modular ontologies: concepts, theories and techniques for knowledge modularization*. Springer, 2009, vol. 5445.
- [255] L. Ora, “Resource description framework (rdf) model and syntax specification,” <http://www.w3.org/TR/REC-rdf-syntax/>, 1999.
- [256] I. Horrocks, “Semantic web: the story so far,” in *Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, 2007, pp. 120–125.
- [257] A.-S. Dadzie and E. Pietriga, “Visualisation of linked data–reprise,” *Semantic Web*, vol. 8, no. 1, pp. 1–21, 2017.
- [258] M. Karnstedt, K.-U. Sattler, and M. Hauswirth, “Scalable distributed indexing and query processing over linked data,” *Journal of Web Semantics*, vol. 10, pp. 3–32, 2012.
- [259] Z. Wu, G. Eadon, S. Das, E. I. Chong, V. Kolovski, M. Annamalai, and J. Srinivasan, “Implementing an inference engine for rdfs/owl constructs and user-defined rules in oracle,” in *2008 IEEE 24th International Conference on Data Engineering*. IEEE, 2008, pp. 1239–1248.
- [260] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, “Open graph benchmark: Datasets for machine learning on graphs,” *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.
- [261] A. Neptune, “Fast reliable graph database built for the cloud,” 2017.
- [262] K. Cagle, “Why the semantic web has failed,” 2016. [Online]. Available: <https://www.linkedin.com/pulse/why-semantic-web-hasfailed-kurt-cagle>
- [263] R. Benjamins, J. Contreras, O. Corcho, and A. Gomez-Perez, “The six challenges of the semantic web,” 2002.
- [264] J. Cabeda, “Semantic web is dead, long live the ai!!!” *Hackernoon, June*, vol. 4, 2017.
- [265] J. Z. Pan, G. Vetere, J. M. Gomez-Perez, and H. Wu, *Exploiting linked data and knowledge graphs in large organisations*. Springer, 2017.
- [266] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

- [267] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [268] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [269] C. Mellish and J. Z. Pan, “Natural language directed inference from ontologies,” *Artificial Intelligence*, vol. 172, no. 10, pp. 1285–1315, 2008.
- [270] E. M. Bender and A. Koller, “Climbing towards nlu: On meaning, form, and understanding in the age of data,” in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 5185–5198.
- [271] C. Zhang, C. Zhang, S. Zheng, Y. Qiao, C. Li, M. Zhang, S. K. Dam, C. M. Thwal, Y. L. Tun, L. L. Huy *et al.*, “A complete survey on generative ai (aigc): Is chatgpt from gpt-4 to gpt-5 all you need?” *arXiv preprint arXiv:2303.11717*, 2023.
- [272] T. Li, M. J. Hosseini, S. Weber, and M. Steedman, “Language models are poor learners of directional inference,” *arXiv preprint arXiv:2210.04695*, 2022.
- [273] Y. He, J. Chen, E. Jiménez-Ruiz, H. Dong, and I. Horrocks, “Language model analysis for ontology subsumption inference,” *arXiv preprint arXiv:2302.06761*, 2023.
- [274] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: a core of semantic knowledge,” in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 697–706.
- [275] G. Weikum, X. L. Dong, S. Razniewski, and F. Suchanek, “Machine knowledge: Creation and curation of comprehensive knowledge bases,” *Found. Tren. Databases*, vol. 10, no. 2-4, pp. 108–490, 2021.
- [276] T. Li, W. Huang, N. Papasrantopoulos, P. Vougiouklis, and J. Z. Pan, “Task-specific pre-training and prompt decomposition for knowledge graph population with language models,” Aug. 2022.
- [277] D. Alivanistos, S. B. Santamaría, M. Cochez, J.-C. Kalo, E. van Krieken, and T. Thanapalasingam, “Prompting as probing: Using language models for knowledge base construction,” Aug. 2022.
- [278] C. De Sa, A. Ratner, C. Ré, J. Shin, F. Wang, S. Wu, and C. Zhang, “Incremental knowledge base construction using DeepDive,” *VLDBJ.*, vol. 26, no. 1, pp. 81–105, Feb. 2017.
- [279] R. Bavishi, C. Lemieux, R. Fox, K. Sen, and I. Stoica, “Autopandas: neural-backed generators for program synthesis,” *Proceedings of the ACM on Programming Languages*, vol. 3, no. OOPSLA, pp. 1–27, 2019.
- [280] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago *et al.*, “Competition-level code generation with alphacode,” *Science*, vol. 378, no. 6624, pp. 1092–1097, 2022.
- [281] X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin, “Deep code comment generation,” in *Proceedings of the 26th conference on program comprehension*, 2018, pp. 200–210.

- [282] B. Li, M. Yan, X. Xia, X. Hu, G. Li, and D. Lo, “Deepcommenter: a deep code comment generation tool with hybrid lexical and syntactical information,” in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1571–1575.
- [283] R. Gupta, S. Pal, A. Kanade, and S. Shevade, “Deepfix: Fixing common c language errors by deep learning,” in *Proceedings of the aaai conference on artificial intelligence*, vol. 31, no. 1, 2017.
- [284] N. Jiang, T. Lutellier, Y. Lou, L. Tan, D. Goldwasser, and X. Zhang, “Knod: Domain knowledge distilled tree decoder for automated program repair,” *arXiv preprint arXiv:2302.01857*, 2023.
- [285] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, “Evaluating large language models trained on code,” *arXiv preprint arXiv:2107.03374*, 2021.
- [286] T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, and L. Jiang, “Sentiment analysis for software engineering: How far can pre-trained transformer models go?” in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 70–80.
- [287] A. M. Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, “Github copilot ai pair programmer: Asset or liability?” *Journal of Systems and Software*, vol. 203, p. 111734, 2023.
- [288] N. Jain, S. Vaidyanath, A. Iyer, N. Natarajan, S. Parthasarathy, S. Rajamani, and R. Sharma, “Jigsaw: Large language models meet program synthesis,” in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1219–1231.
- [289] N. Nguyen and S. Nadi, “An empirical evaluation of github copilot’s code suggestions,” in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 1–5.
- [290] P. Vaithilingam, T. Zhang, and E. L. Glassman, “Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models,” in *Chi conference on human factors in computing systems extended abstracts*, 2022, pp. 1–7.
- [291] B. T. Hammond Pearce, B. Ahmad, R. Karri, and B. Dolan-Gavitt, “Can openai codex and other large language models help us fix security bugs,” *arXiv preprint arXiv:2112.02125*, 2021.
- [292] S. MacNeil, A. Tran, A. Hellas, J. Kim, S. Sarsa, P. Denny, S. Bernstein, and J. Leinonen, “Experiences from using code explanations generated by large language models in a web software development e-book,” in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, pp. 931–937.
- [293] B. Chen, F. Zhang, A. Nguyen, D. Zan, Z. Lin, J.-G. Lou, and W. Chen, “Codet: Code generation with generated tests,” *arXiv preprint arXiv:2207.10397*, 2022.
- [294] Z. Fan, X. Gao, M. Mirchev, A. Roychoudhury, and S. H. Tan, “Automated repair of programs from large language models,” in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 1469–1481.

- [295] N. M. S. Surameery and M. Y. Shakor, "Use chat gpt to solve programming bugs," *International Journal of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290*, vol. 3, no. 01, pp. 17–22, 2023.
- [296] D. Sobania, M. Briesch, C. Hanna, and J. Petke, "An analysis of the automatic bug fixing performance of chatgpt," *arXiv preprint arXiv:2301.08653*, 2023.
- [297] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," *arXiv preprint arXiv:2305.10601*, 2023.
- [298] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 716–23 736, 2022.
- [299] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pretrained language model for scientific text," *arXiv preprint arXiv:1903.10676*, 2019.
- [300] E. Mashhadi and H. Hemmati, "Applying codebert for automated program repair of java simple bugs," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 2021, pp. 505–509.
- [301] H. Tian, K. Liu, A. K. Kaboré, A. Koyuncu, L. Li, J. Klein, and T. F. Bissyandé, "Evaluating representation learning of code changes for predicting patch correctness in program repair," in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, 2020, pp. 981–992.
- [302] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.
- [303] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.
- [304] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [305] C. Le Goues, M. Pradel, A. Roychoudhury, and S. Chandra, "Automatic program repair," *IEEE Software*, vol. 38, no. 4, pp. 22–27, 2021.
- [306] M. Motwani, M. Soto, Y. Brun, R. Just, and C. Le Goues, "Quality of automated program repair on real-world defects," *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 637–661, 2020.
- [307] S. Gulwani, I. Radiček, and F. Zuleger, "Automated clustering and program repair for introductory programming assignments," *ACM SIGPLAN Notices*, vol. 53, no. 4, pp. 465–480, 2018.
- [308] Y. Hu, U. Z. Ahmed, S. Mechtaev, B. Leong, and A. Roychoudhury, "Re-factoring based program repair applied to programming assignments," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 388–398.
- [309] S. Chakraborty, Y. Ding, M. Allamanis, and B. Ray, "Codit: Code editing with tree-based neural models," *IEEE Transactions on Software Engineering*, vol. 48, no. 4, pp. 1385–1399, 2020.

- [310] H. Ye, M. Martinez, and M. Monperrus, "Neural program repair with execution-based backpropagation," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1506–1518.
- [311] H. Pearce, B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, "Examining zero-shot vulnerability repair with large language models," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 2339–2356.
- [312] J. A. Prenner, H. Babii, and R. Robbes, "Can openai's codex fix bugs? an evaluation on quixbugs," in *Proceedings of the Third International Workshop on Automated Program Repair*, 2022, pp. 69–75.
- [313] C. S. Xia and L. Zhang, "Keep the conversation going: Fixing 162 out of 337 bugs for \$0.42 each using chatgpt," *arXiv preprint arXiv:2304.00385*, 2023.
- [314] I. Zakori. (2023) Gpt-4: The world's most powerful ai model - an analysis of its performance decline. Accessed: 11 August 2023. [Online]. Available: <https://artificialcorner.com/gpt-4-the-worlds-most-powerful-ai-model-an-analysis-of-its-performance-decline-598ad723a595>
- [315] R. Madaan. (2023, 7) Chatgpt gpt-4 response quality and performance drop issues to be investigated. Accessed: 11 August 2023. [Online]. Available: <https://piunikaweb.com/2023/07/14/chatgpt-gpt-4-response-quality-performance-drop-issues/>
- [316] A. Barr. (2023, 7) Openai's gpt-4 ai model got lazier and dumber - chatgpt. Accessed: 11 August 2023. [Online]. Available: <https://www.businessinsider.com/openai-gpt4-ai-model-got-lazier-dumber-chatgpt-2023-7?r=US&IR=T>
- [317] L. Chen, M. Zaharia, and J. Zou, "How is chatgpt's behavior changing over time?" *arXiv preprint arXiv:2307.09009*, 2023.
- [318] T. Boland and P. E. Black, "Juliet 1.1 c/c++ and java test suite," *Computer*, vol. 45, no. 10, pp. 88–90, 2012.