

# New Advances in Tamper Evident Technologies



Ehsan Toreini  
School of Computing  
Newcastle University

A thesis submitted for the degree of  
*Doctor of Philosophy*

2017

## Acknowledgements

I would like to thank everyone who warm-heartedly supported me during my PhD studies. I truly thank my supervisor, Prof. Feng Hao, for his endless support and inspiration through this journey. He encouraged me to pursue my research interest and not to give up. I thank my second supervisor, Dr. Siamak Fayyaz Shahandashti, for his friendship and support during years. He taught me lessons that “the devil is in the detail”. I also thank my examiners, Dr. Changyu Dong and Dr Steven J. Murdoch for their constructive feedback.

My dissertation would not be complete without naming my supporting colleagues and friends. I thank Prof. Brian Randell and Prof. Feng Hao for trusting me in the first place when I arrived at Newcastle University. I thank Prof. Aad Van Moorsel, and the SRS group at the School of Computing, for the world-class environment that they provided to me. My colleagues: Sami Alajrami, Razgar Ebrahimi, David Ebo Adjepon-Yamoah, Roberto Metere, and Peter Carmichael have all been like a second family at work. I specially thank my friends beyond the university: Mehran, Nasrin, Ramin, Elham, Zoya, Nadia, and Fesenjoon for their encouragement over the years.

I truly thank my family who supported me unconditionally throughout this journey. I thank my adorable parents, Moslem and Roya, who taught me how to live and enjoy life. I wish to specially thank my uncle and his wife, Mokhtar and Lynn, for their endless support during my studies. I thank our dear daughter, Maya, who gave me the last push toward the end line. Last but not least, this thesis could not be finished without the help and support of my lovely wife, Dr. Maryam Mehrnezhad. She has been my partner, my friend, my colleague, and my advisor at the same time.

## Abstract

Tampering is a thousands-years-old problem. Ancient Mesopotamian civilizations developed mechanisms to detect tampering of their purchase receipts on clay tablets. Today, the advances in the technology have equipped adversaries with more modern techniques to perform attacks on physical items (such as banknotes and passports), as well as cyber products (software and webpages). Consequently, tampering detection mechanisms need to be developed as new attacks emerge in both physical and cyber domains. In this dissertation, we divide our research into two parts, concerning tampering in physical and in cyber domains respectively. In each part, we propose a new method for tampering detection.

In the first part, we propose a novel paper fingerprinting technique based on analysing the translucent patterns revealed when a light source shines through the paper. These patterns represent the inherent texture of paper, formed by the random interleaving of wooden particles during the manufacturing process. We show these patterns can be easily captured by a commodity camera and condensed into to a compact 2048-bit finger-print code. Prominent works in this area (Nature 2005, IEEE S&P 2009, CCS 2011) have all focused on fingerprinting paper based on the paper “surface”. We are motivated by the observation that capturing the surface alone misses important distinctive features such as the non-even thickness, the random distribution of impurities, and different materials in the paper with varying opacities. Through experiments, we demonstrate that the embedded paper texture provides a more reliable source for fingerprinting than features on the surface. Based on the collected datasets, we achieve 0% false rejection and 0% false acceptance rates. We further report that our extracted fingerprints contain 807 degrees-of-freedom (DoF), which is much higher than the 249 DoF with iris codes (that have the same size of 2048 bits). The high amount of DoF for texture-based fingerprints makes our method extremely scalable for recognition among very

large databases; it also allows secure usage of the extracted fingerprint in privacy-preserving authentication schemes based on error correction techniques.

In the second part, we address an important real-world problem: how to ensure the integrity of delivering web content in the presence of man-in-the-browser (MITB) attacks by malicious web extensions? Browser extensions have powerful privileges to manipulate a user’s view of a web page by modifying the underlying Document Object Model (DOM). To demonstrate the threat, we implement two attacks on real-world online banking websites (HSBC and Barclays) and show how a malicious extension can covertly compromise the user’s bank accounts. To address this problem, we propose a cryptographic protocol called DOMtegrity to ensure the end-to-end integrity of a web page’s DOM from delivering at a server to the final display in a client’s browser. The novelty of our solution lies in exploiting subtle differences between browser extensions and in-line JavaScript code in terms of their rights to access WebSocket channels, as well as leveraging the latest Web Crypto API support added in modern browsers. We show how DOMtegrity prevents the earlier attacks and a whole range of man-in-the-browser attacks that involve maliciously changing the DOM structure of a web page. We conduct experiments on more than 14,000 real-world extensions to evaluate the effectiveness of DOMtegrity and its compatibility with existing extensions. To the best of our knowledge, DOMtegrity is the first solution that effectively protects the integrity of DOM against malicious extensions without needing to modify the existing browser architecture or requiring extra hardware.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Contributions . . . . .	3
1.3	Thesis Outline . . . . .	4
<b>2</b>	<b>Tamper Evident Technologies</b>	<b>5</b>
2.1	Overview . . . . .	5
2.2	Tamper Evident Security Considerations . . . . .	7
2.3	Physical Tamper Evident Protections . . . . .	9
2.3.1	Unique Objects . . . . .	10
2.3.2	Physical Unclonable Functions . . . . .	12
2.3.3	Security Printing . . . . .	17
2.4	Cyber Tamper-Evident Solutions . . . . .	19
2.4.1	Source Code Protection . . . . .	20
2.4.2	Other Aspects . . . . .	23
2.4.2.1	Copy Protection Mechanisms . . . . .	23
2.4.2.2	Digital Media Protection . . . . .	24
2.4.2.3	Software Packaging . . . . .	25
2.5	Summary . . . . .	25
<b>3</b>	<b>Physical Tamper Evidence:</b>	
	<b>Paper Sheet Case</b>	<b>27</b>
3.1	Introduction . . . . .	29
3.2	Paper Texture . . . . .	30
3.3	Related work . . . . .	32
3.4	Texture to the Rescue: Practical Paper Fingerprinting based on Texture Patterns . . . . .	35
3.4.1	Preparation Phase . . . . .	36

3.4.2	Gabor Filter . . . . .	38
3.4.3	Fingerprint Generation . . . . .	39
3.4.4	Fractional Hamming Distance . . . . .	39
3.5	Evaluation . . . . .	40
3.5.1	Parameter Settings & Experiment Configurations . . . . .	40
3.5.2	Evaluation Framework . . . . .	42
3.5.2.1	Space Dimension . . . . .	45
3.5.2.2	Time Dimension . . . . .	45
3.5.2.3	Device Dimension . . . . .	46
3.5.3	Reflection vs. Transmission . . . . .	47
3.5.4	Determining Gabor Scale & Orientation . . . . .	49
3.5.5	The Benchmark Dataset . . . . .	50
3.5.6	Experiment Results . . . . .	52
3.6	Robustness Analysis . . . . .	56
3.6.1	Impact of Non-Ideal Data Collection . . . . .	56
3.6.2	Impact of Non-Ideal Paper Handling . . . . .	57
3.6.3	Impact of a Different Light Source . . . . .	59
3.7	Authentication Protocols . . . . .	62
3.7.1	Trust Assumptions . . . . .	62
3.7.2	Comparison Based on Hamming Distance . . . . .	63
3.7.3	Paper Fingerprint Encryption . . . . .	65
3.8	Media Coverage . . . . .	69
3.9	Summary . . . . .	70

<b>4</b>	<b>Cyber Tamper Evidence:</b>	
	<b>Web Page Case</b>	<b>71</b>
4.1	Introduction . . . . .	72
4.2	Related Work . . . . .	73
4.3	Malicious Extension Attacks on Online Banking . . . . .	76
4.3.1	WebExtensions Capabilities . . . . .	76
4.3.2	HSBC Attack . . . . .	77
4.3.3	Barclays Attack . . . . .	78
4.4	Our Proposed Solution: DOMtegrity . . . . .	81
4.4.1	WebExtensions Security Model . . . . .	81
4.4.2	Design Overview . . . . .	82
4.4.3	Detailed Description . . . . .	85

4.4.4	How DOMtegrity Prevents Attacks . . . . .	88
4.5	Discussion on Impersonation . . . . .	89
4.6	Implementation and Evaluation . . . . .	93
4.6.1	Confirming DOMtegrity Effectiveness . . . . .	94
4.6.2	Performance Evaluations . . . . .	95
4.6.3	Compatibility with Real-world Extensions . . . . .	97
4.7	Further Discussion . . . . .	99
4.8	Industry Acknowledgement . . . . .	102
4.9	Summary . . . . .	102
<b>5</b>	<b>Conclusion and Future Work</b>	<b>103</b>
5.1	Summary . . . . .	103
5.2	Future Work . . . . .	105
<b>Bibliography</b>		<b>106</b>

# List of Figures

2.1	Examples of real-world physical tamper evident technology practices.	7
2.2	Metal strip on a British pound banknote. Forgers managed to apply a combination of metallic patches and painting to counterfeit the older version of such banknotes successfully. Presentation of the vulnerable banknote is not possible due to legal restrictions. . . . .	9
3.1	The surface and texture of the same area of a paper sheet as captured by a camera based on a) reflective and b) transmissive light. . . . .	30
3.2	Step-by-step rotation recognition process in the preparation phase. . .	36
3.3	Gray code for a complex value $m_{ij} = a + bi$ in the complex plain. . .	40
3.4	The equipment used in our experiments. . . . .	42
3.5	Capturing a photo, in case of (a) transmission, and (b) reflection, using the same digital camera and light source. . . . .	47
3.6	Hamming distance distributions for surface and texture. . . . .	48
3.7	Results of (a) decidability and (b) degrees of freedom in scales 1 to 7 and orientations 1 to 8. . . . .	51
3.8	Hamming distance distributions in the benchmark dataset. . . . .	52
3.9	Histograms of Hamming distances in the benchmark dataset. . . . .	54
3.10	The captured photo under near-ideal and non-ideal situations. . . . .	56
3.11	The Hamming distance distributions for robustness experiments. . . .	60
3.12	Distributions of HDs for the light box experiment. . . . .	61
3.13	Generated QR Code in the authentication protocol. This QR code contains the encrypted fingerprint, $H(k)$ and a digital signature for both items resulting into 718 bytes (size varies depending on the type of implemented cryptographic operation). . . . .	66
3.14	Histogram of Hamming distances between raw fingerprints without masks. . . . .	69

4.1	The HSBC customer service page modified by the malicious extension to contain a message indicating website technical difficulties. . . . .	78
4.2	The Barclays instructions page modified by the malicious extension to include the attacker’s account number (redacted as XXXXXXXX) as the REF number. . . . .	79
4.3	The Internet and Chrome zones of a modern browser and how web pages, extensions, and plug-ins interact [4] . . . . .	82
4.4	sequence diagram for DOMtegrity . . . . .	84
4.5	An overview of document.pid and the inability of extensions to modify this region of DOM. . . . .	86
4.6	The interval between creation of Websocket objects in the extension and pid.js versus the interval that the opening handshake requests sent from the browser. Solid lines indicate regression of delay in each corresponding interval when extension injects commands to delay pid.js Websocket’s execution. . . . .	92
4.7	The time when pid.js’s opening handshake dispatch versus when blocking executed in the extension. Both numbers are measured once malicious extension’s Websocket is opened successfully. Solid lines indicate linear regression of in each series when extension injects commands to delay pid.js Websocket’s execution. . . . .	94
4.8	Box plots of elapsed times for PID and HMAC calculations in 100 executions in Chrome and Firefox. . . . .	97
4.9	Examples of source code modifications during parsing in browsers . . . . .	100

# List of Tables

3.1	False recognition rates of all datasets considering a fractional HD threshold of 0.4 . . . . .	55
3.2	PUF metrics for all datasets and two typical PUFs . . . . .	55
3.3	Impact of Robustness Experiments on PUF metrics . . . . .	55
3.4	False Acceptance Rate (FAR) for comparing two fingerprints . . . . .	64
4.1	Capabilities of extension and in-line script (W3C [214]) . . . . .	83
4.2	Capabilities of extension and in-line script before and after Chrome v.58. . . . .	90
4.3	Average elapsed times for stopping event propagation in Chrome and Firefox for our experimental web pages . . . . .	96
4.4	Average and standard deviation of the elapsed times on the server side for 100 executions of each step of the protocol . . . . .	97
4.5	Likelihood of mutation type occurrence among tested rejected extensions for each mutation type . . . . .	99

# Chapter 1

## Introduction

### 1.1 Motivation

Tamper evident technologies are as old as the economy. They have evolved as the commerce progressed. From ancient times to the present, producers and service providers have been trying to create a *tampering-aware* environment [187] for their products in order to prevent physical tampering. Nowadays, tamper evident applications cover areas like brand protection, banknote and coinage security, sealing protection, passport security, document anti-forgery and anti-counterfeiting packaging.

There have been various classical ways to protect a *physical* product against tampering. The common purpose of these methods is to add a unique property to the product in order to safeguard it against forgery and counterfeiting. In recent years, the unique inherited features of a product (also known as substrate properties [5]) have been investigated, too. These features are extremely hard to clone since they are randomly shaped during the manufacturing process [199]. In an effort to advance in this field, Pappu et al. [156] proposed a cryptographic method to quantify the inherited characteristics into a robust unique identifier to track the product.

In the digital era, the research and practice to prevent tampering extends to *cyber* products (such as software), too. The need to protect software against possible forgeries seems inevitable since hacking is common in the IT field.

Protection of cyber products against tampering usually does not have physical dimensions and features. Cryptographic primitives have been widely used to build protection mechanisms in order to ensure integrity, authenticity and security of the delivery and execution of a software product to the end-users.

The current tamper protection mechanisms are not bullet-proof though. In the EU alone, counterfeit products amount up to 85 billion EUR annually, which equals 5% of all EU imports [147]. Business Software Alliance (BSA), a trade group that represents

a number of world's largest software makers, estimated the cost of software piracy is about 9.1 billion USD only in the United States, 17% of all software purchases in the US [30].

Other types of tampering crimes are on the rise, too. Valuable items such as banknote, passport and other identity-related documents are facing the same threats. Bank of England officially announced that the total number of detected counterfeit banknotes in 2016 was 347000 (valued around £7.47 million pounds). The number of the removed banknotes before entering financial transaction circulation was 6000. In the first half of 2017, 237000 counterfeit banknotes have been reported (valued more than £4.851 million pounds), all of them managed to enter financial transaction circulation [13]. Also, the number of attempts to enter the UK with fake passports has been rising from 2012 to 2015 [25]. The Syrian civil war has significantly increased the fraudulent passport related crimes for immigration in Europe, UK and USA in the past few years [194, 162].

Technological developments provide more opportunities for tampering attacks. High-quality commodity printing devices available at affordable prices, in conjunction with freely available software tools to manipulate digital images, encouraged amateurs to commit counterfeiting and forging. These exercises make detection more challenging since they are ah-hoc and geographically distributed. [5].

Meanwhile, cyber-security related crimes are on the rise too. Pricewaterhouse Coopers (P.W.C.), one of the world's leading business consultants, reported a sharp rise in these crimes from 2016 to 2017, jumping from 4th to 2nd place among the most-reported types of economic crimes [165]. Official surveys for England and Wales announced 3.4 million cases of computer fraud in 2017 alone [54].

Meanwhile, the tampering aware solutions need to develop as different types of attacks evolve. Hence, the need for new proposals for more modern tampering protection solutions is urgent. This dissertation concentrates on developing new techniques to detect tampering in both the physical and cyber domains.

In the physical domain, we focus on preventing the counterfeiting of a paper document. It is especially an interesting case study because the paper sheets are still widely utilized in our everyday life, e.g. legal documents, paper forms, banknotes and certificates. In fact, recent reports by De La Rue (leading corporation in printing banknote and passports worldwide) predicted that demand for banknotes to rise by 3-4% a year in the foreseeable future [193]. This happens in spite of the rise in digital money and crypto-currencies like Bitcoin.

In the cyber domain, we propose a solution to detect modifications of a web page by malicious browser extensions. Browser extensions enjoy more privileges than regular web pages in the browser. Consequently, clandestine modification of the web page is possible without being detected. Previous research suggests various types of defensive mechanisms to contain malicious extensions; however, they did not deter the extension’s threat completely. Kaspersky, McAfee and Symantec included malicious extensions as a top online threat in their annual security reports in 2016 [189, 106, 96].

## 1.2 Contributions

In this section, we have summarized our contributions in this dissertation.

1. We provide an overview of the existing approaches in tamper-evident solutions both in the physical and cyber domains. In each domain, we categorize the research and practice and discuss the challenges.
2. In detecting tampering in the physical domain, we contribute the following:
  - We revisit paper fingerprinting and propose to use the textural patterns revealed by passing light through a paper sheet as a reliable source for extracting a fingerprint, as opposed to previous measures which are based on paper surface imperfections.
  - We design an efficient paper fingerprinting algorithm based on error correction code and image processing techniques, and carry out experiments to show that our method can be used to efficiently extract a reliable and unique fingerprint using a photo taken by an off-the-shelf camera. Our proposed method is feasible and inexpensive to deploy in practice.
  - We conduct further experiments to demonstrate that our paper fingerprinting method is robust against: (a) non-ideal photo capturing settings such as when the paper is rotated and the light source is changed, and (b) non-ideal paper handling situations such as crumpling, soaking, heating and scribbling on the surface.
3. In detecting tampering in the cyber domain, our contributions are as follows:
  - We propose DOMtegrity, a cryptographic protocol to protect end-to-end integrity of a web page’s DOM from the point of delivery at a server to the

final display in a client’s browser. This is the first solution that works with the standard browser extension architecture without needing any external hardware.

- We present an efficient implementation of DOMtegrity, using JavaScript on the client side and Node.js on the server side, and demonstrate that the proposed solution is efficient and only adds a small overhead to the computation load and communication bandwidth.
- As part of the evaluation, we implement two attacks on real-world online banking systems (HSBC and Barclays) to show how a malicious extension can compromise the security of the user’s bank account, and how DOMtegrity can prevent such attacks as well as other man-in-the-browser (MITB) attacks that involve maliciously changing the DOM structure of a web page.

### 1.3 Thesis Outline

in Chapter 2, we will discuss the state-of-the-art research and practice in tamper evident solutions. We categorize the contents of this chapter into two main classes: physical and cyber domains. In each domain, we will review the existing solutions and discuss the challenges.

In Chapter 3, we will discuss our proposed tamper evident solution for paper fingerprinting. We will review the previous research, and explain their strengths and shortcomings. We will introduce our proposed solution and provide a robust evaluation framework. Finally, we will perform extensive experiments to support our proposed method and analyse different security aspects of it.

In Chapter 4, we will introduce our solution for web integrity. We will present an introduction to the previous literature. Next, we will propose our protocol to ensure the integrity of a web page in a client’s browser in the presence of a malicious browser extension. We name our solution “*DOMtegrity*”. Finally, we will evaluate effectiveness of DOMtegrity against malicious extensions in different attack scenarios.

In Chapter 5, we will conclude this dissertation and suggest future research.

# Chapter 2

## Tamper Evident Technologies

In this chapter, we discuss the current methods of providing tamper evidence in physical and cyber domains. In physical tamper evidence, we identify three trends in current research: Unique Objects, Physical Unclonable Objects and Security Printing. In the cyber tamper evidence, we overview software source code protection mechanisms in detail and, then discuss other trends in providing tamper evidence for software.

### 2.1 Overview

**Definition.** Oxford dictionary defines the word *tamper* as “to meddle or interfere with (a thing) so as to misuse, alter, corrupt, or pervert it” [152]. These modifications usually occur insidiously by individuals, rival organizations or hostile governments. Also, the attacker (tamperer) does not have permission to perform such alterations. These modifications can happen in any stage of the product’s life cycle, i.e. providing the production materials, manufacturing process, retails and consumption [119].

Generally, *forgery* is the illegal act of producing a fabricated product with the purpose of convincing a victim of its authenticity. Criminals intend to implement forgery in various illegitimate ways such as counterfeiting and tampering. However, one needs to clarify the distinction between these terms. Products sidetracked from their proper distribution channel, or sold past their expiry date, or by modification of the package are associated with counterfeiting [148]. Counterfeits are unauthorized *reproductions* of a trademarked brand, which are closely similar or identical to genuine articles [52]. The core strategy of counterfeiting is based on “a fake (typically a crude one) is made outside of the manufacturing plant using new seals, used parts or completely original materials [177]”. In contrast, tampering would solely concentrate on *modification* of an authentic product for the purpose of forgery-related crimes. A

solution might protect a product from counterfeiting, tampering or both at the same time; however, in the context of this dissertation, we would address only tampering aspect of our proposed solutions.

Protection against tampering involve either tamper evident or tamper resistant methods. The former reveals modifications and the latter prevents them. In this dissertation we will focus on *tamper evident* technologies.

Tamper evident is a property which is “designed to make obvious any improper interference with a product (esp. food stuff or medicine) before sale” [153]. Tamper evident, a.k.a. tamper-indicating, solutions are mechanisms that disclose any malicious modification to an object. These solutions have direct application in sealing and secure packaging (such as the pharmaceutical product packaging, see Figure 2.1(a)). They are also exploited in printing and anti-counterfeiting solutions (such as banknotes and passport protection, see Figure 2.1(b)).

Furthermore, protecting a product from forging requires a mechanism to verify whether the product is authentic or not (*authentication*). One of the common methods to provide such verification is to consider one or more unique, unclonable features for the product (*fingerprints*) [198]. The fingerprint can be embedded or attached to the product. Tamper evident solutions would protect the integrity of the fingerprint by preventing attackers to profit by either repacking or reselling of the products [177]. In the context of this dissertation, we would propose solutions to verify the authenticity and integrity of such fingerprint in both physical and cyber domains.

The growth of digital information has led to the employment of tamper evident application to software products, too. Information technology companies require to protect their data, source code and packaging against cyber forms of tampering. Mechanisms to protect software against piracy (like licensing methods), file modifications (like digital watermarks) and network tampering (like TLS/SSL protocol) are all real-world instances of cyber tamper evidence technologies.

Many products that we use in everyday life need to be protected against tampering. However, since complete tamper proofing is extremely difficult to achieve [5, p. 434], more solutions are focused just on making these items tamper evident. This seems to be a more realistic approach since it fulfils such protection in a less sophisticated way.

Tamper evident solutions need constant adaptation to the latest tampering attacks. Investment in tamper evident solutions is specially beneficial to producers and service providers because counterfeiting products impose heavy revenue losses to them. Forgery related crimes account for as much as \$461 billion annually worldwide



(a) Tamper evident sealing of a medical product.  
 (b) Anti-counterfeiting solutions printed on a polymer £5 banknote.

Figure 2.1: Examples of real-world physical tamper evident technology practices.

in 2013, up to 2.5% of the total world trade [147]. Hence, for instance, the banknote printing system gets updated every few years to make the banknotes resistant against new trends of such crimes [5, p. 441].

## 2.2 Tamper Evident Security Considerations

**Threat Model.** IBM classifies attackers to a tamper-aware solution based on the level of experience and access to sophisticated equipment. This classification has been first proposed in the design documentation of their tamper resistant chip, IBM 4758 crypto-processor [1], as follows:

- **Clever Outsiders:** they are intelligent people with limited knowledge about the design. Their access to sophisticated equipment is also limited. Most of the times, they try to leverage the shortcomings in the design.
- **Knowledgeable Insiders:** they have enough knowledge and experience to defeat the tamper evident design; however, they do not have complete understanding of the system. Since they are insiders, they have potential access to many parts of the solution. They are equipped with sophisticated instruments and analysis power.
- **Funded Organizations:** they are able to systematically target the design with extensive support from experts. They have enough funding and resources to perform the most advanced analysis of the system. Furthermore, They can recruit knowledgeable insiders as a part of their team.

Each tamper evident design aims to target protection against a subset of the above attackers. The Federal Information Processing Standard (FIPS) provides a certification scheme to measure the degree of protection in secure chip designs [39]. They have four levels of protection standard. Level 4 is the most secure design that protects users from funded organization.

**Inspections.** Every tamper evident solution requires a detailed inspection framework, too. Solution designers should consider different methods to help an inspector spot tampering seamlessly. For example, examining a banknote watermark usually aims for amateur inspector whereas the embedded UV patterns in a banknote could help more careful investigation.

The inspection of a product is a balanced approach toward time, budget, equipment and accuracy. The inspector's personality influences the level of engagement in the inspection, too. Also, the inspectors vary widely based on their motivation. For example, a supermarket salesman who inspects a banknote, usually tends to accept it and thus, he performs the inspection more carelessly than a bank clerk [5, p. 437].

Van Renesse [210] divided tampering inspection into three categories:

1. **Primary:** This type of inspection is performed by an inexperienced, untrained person. In most cases, the motivation for inspection is to accept an unreliable item. Thus the inspector tends to perform a slight quick observation and possibly would decide to pass it.
2. **Secondary:** This type of inspection is performed by experienced, motivated person e.g. a bank teller to vet a suspicious banknote. He might possess a special equipment to inspect the item. However, his equipment would be limited in value and bulk. Usually, criminals target secondary-level inspections [5, p. 437].
3. **Tertiary:** This category is only performed in special laboratories such as manufacturer's or governmental offices. The designers of the tamper-evident technologies might be available for consultation. They are equipped with sophisticated and modern inspection equipment.

**Solution Design.** Designers should realistically assess the purpose and limitations of a tamper evident solution. There have been a lot of occasions for overbelieving in a tamper evident solution. For example, the British banknote designers in the 90s introduced a metal strip passed over and through the banknote paper body. They assumed their proposed solution was invincible. However, criminals managed to create a perception of the same approach by pasting metal pieces on a banknote



Figure 2.2: Metal strip on a British pound banknote. Forgers managed to apply a combination of metallic patches and painting to counterfeit the older version of such banknotes successfully. Presentation of the vulnerable banknote is not possible due to legal restrictions.

surface and painting the rest. At the time that the criminals were caught, they already managed to produce tens of millions of pounds worth of notes over a period of several years [49], see Figure 2.2.

No single solution is enough to provide a bullet-proof tamper evidence; therefore, usually multiple approaches are implemented together to reduce the risk of a successful attacker. However, such strategy can only extend the amount of time required for a successful tampering attack. Meanwhile, the designers should focus on updating their solutions before the attackers would be able to defeat them.

In this thesis, we consider the case of adversaries that are either *clever outsider* or *knowledgeable insider* with access to *primary* and *secondary* inspection equipment.

## 2.3 Physical Tamper Evident Protections

In this section, we explain the new trends in physical tamper evident solutions. They are designed to physically protect a product from malicious modifications. Physical tamper evidence solutions date back to Sumerian civilization in around 3000 B.C.

In recent years, a new tamper evident approach has emerged in the literature that focuses on the inherent, random properties of physical objects as unclonability measure. The two more important classes of such disorder-based security systems are *Unique Objects (UNOs)* and *Physical Unclonable Functions (PUFs)*.

On the other hand, modern tamper evident solutions in the industry are a combination of advanced physical and chemical procedures. The most prominent set of solutions is regarded as *security printing*. These techniques enable industrialized manufacturing of tamper evident seals, secure documents and packagings. Security

printing is an expensive process that requires special equipment. Its costly procedure makes it economically profitable when it is exploited in large scales, since it makes counterfeiting infeasible [207].

In recent years, the industry's focus on 3D printing enabled the embedding of inherent properties to a product [122]. So far, such efforts have not been fruitful in larger commercial scales yet. Moreover, the tamper evident sealing faces existential threat since the current solutions are too easy to defeat [101]. “Electronic seals” mean to preserve the reliable features of current sealing while adding electronic components to overcome the weaknesses. Jackson [101] argues that the most important downside of traditional sealing is the lack of detection of unauthorized access, as he called “alarm condition”. The combination of security printing techniques, secure storage of cryptographic keys and unique properties of seals is likely to shape into next generation of stronger packaging solutions. The increasing number of start-ups active in PUF and anti-counterfeiting technologies as well as the increasing interest of venture capitalists to invest on them, suggests we should expect to see more practical evolutions in current tamper-evident technologies in our everyday lives.

### 2.3.1 Unique Objects

A Unique Object is a physical system that shows a set of small, persistent analogue properties that can be measured by external equipment. Such properties would not be similar to any other samples of the same object [170]. They are also addressed as *fingerprint* of the object in the literature.

Rührmair et al. [170] discussed a unique object should have the following properties:

1. *Disorder*: The fingerprint should be based on the unique disorder that belongs only to the object itself.
2. *Operability*: The fingerprint must be sufficiently persistent over time. Therefore, it should be robust against ageing and some environmental hardships. In other words, it should be repeatable in different circumstances.
3. *Unclonability*: It must be extremely expensive or impractical for an entity (manufacturer, governments and more common bodies with limited resources) to produce another object with the exact properties as the unique object.

**Types of UNOs.** One well-known category of UNOs is called *Sprayed Random Surfaces*. In this approach, a unique liquid substance is sprayed on an object and

rests until it is solid. The hardened substance shapes a random unique pattern that is considered as the fingerprint of the object. Inspectors use these unique random patterns on the object for verification.

Sprayed random surfaces have been used to protect nuclear weapon treaties during the cold war [170, 73]. The inspectors sprayed a layer of light reflective material on the nuclear missiles. When exposed to light, the random scattering of light from the surface of this substance would generate a unique pattern. During inspections, a trusted device was used to measure these illuminations and verify whether the missile is genuine or not. Such measuring was performed under strict regulations and supervision of different international authoritative bodies and therefore, they did not consider attacks such as impersonation during the remote authentication of the scattered surface in their threat models.

Another category of UNOs is regarded as *Fibre-Based Random Surfaces* which relies on randomly distributed fibres in a solid-state object. For example, bank cards carry a thin substrate that is measured by a magnetic reader [26]. Simmons [179] were the first to suggest combination of fibre-based unique objects and digital signature for off-line verification of the labels. The authors of [183, 140] proposed the creation of unforgeable postal stamps and document authentication from random fibres and digital signatures.

The use of randomly scattered fibres inside a fixing matrix is another approach in fibre-based random surface solutions [60, 109, 42, 33]. One example involves scattering light-sensitive objects inside a transparent gluing material, so that they can be fixed to the object. When illuminated, the fibres would light up and their locations would be the unique feature by which the object is verified with [109, 42].

**Applications of UNOs.** The most prominent application of UNOs is to label valuable items such as banknotes, passports, bank cards, access tokens, etc. [170]. Such protection against tampering would provide a ground to authenticate them for different purposes such as anti-counterfeiting or tracking.

Moreover, these labels could ensure if a product is genuine , and so provide brand protection and reduce losses caused by counterfeiting. This solution can be implemented by various mechanisms [170], as follows:

- *Centralized Verification:* The product contains measurable characteristics either by attaching a unique object or using its own inherited properties. This fingerprint is stored in a central database. Later, it can be verified by cross comparison between the newly measured fingerprint and the ones stored in the database.

- *Certificate of Authority (COA)*: This method has been proposed in [179, 60, 111]. In this approach, the fingerprint information is stored directly on the item, ie. via a printed barcode. The COA includes numerical encoding of the fingerprint, error-correction codes, item-related information (such as the origin of the producer of the item) together with a digital signature of all the information. Verification is done off-line by measuring the fingerprint with a trusted device and then checking whether the two fingerprints match. The verifier should check the validity of the digital signature to ensure the authenticity of other information printed via the barcode.

UNOs have security features that make them important in providing tamper evidence. The first feature is by nature, UNOs have no secret. Everything related to them is public; however, unclonable. On the other hand, UNOs are structurally sensitive to modifications. The unique measured fingerprint from a UNO is affected by alternations to its inner structure [170].

In the literature, the concept and terms unique objects and non-electric PUF (which will be discussed later) are used interchangeably. The dominant approach is to consider UNOs as a type of PUF [132]. On the other hand, some researchers consider PUF only electronic so they differentiate between the two [170]. In our thesis we prefer to follow the dominant approach and consider UNOs as a type of PUF.

### 2.3.2 Physical Unclonable Functions

It is becoming increasingly important to authenticate physical objects. The universal-spread network of computers, mobiles, tablets and now more common *things* is a significant part of everyone's ordinary life and thus, accurate identification of all these connected objects is important. In the same way as humans, objects can be authenticated based on their physical characteristics. Such authentication mechanism would distinguish genuine objects from tampered or counterfeit ones.

The first method for measuring unique object features for purpose of authentication was proposed by Wienser in late 1960's [18, 222]. He suggested an anti-counterfeiting method that utilized the no-cloning theorem of quantum physics. This theorem proves that it is impossible to duplicate an unknown quantum state with a high probability of success. He suggested to equip an item with quantum system in a way that is only known to the issuer. Consequently, one can be confident that an attacker is unable to clone the object, but the issuer would still be able to verify

its authenticity. His proposed method was theoretically correct; however, it is not possible to maintain a persistent quantum state for a long time [199].

In 1991, a revolutionary idea was proposed by Simmons [180]. Previously, the anti-counterfeiting marks embedded inside an object were manufactured in the same size. Simmons argued it would not protect the object against tampering and counterfeiting since the protection mechanism is predictable. Instead, he suggested any solution needs to consider unique, irreproducible physical features. He also proposed to use digital signatures to sign the record of random physical features.

In 2001, Pappu [159] proposed the idea of PUF for the first time. Physical Unclonable Functions (PUFs) are physical objects that are inherently unclonable because they contain many random components that are extremely hard to replicate. Other terminologies such as Physical Random Functions or Physical One-Way Functions are used to refer to PUFs as well [199]. They are also called Challenge Response Pairs (CRP) in the literature [199].

PUFs map *challenges* to *responses* [199]. A challenge is the action that invokes the PUF and a response is the reaction that is received from it. This reaction, however, is subject to noise and is slightly different each time. The term *function* in PUF refers to the idea that such response is the result of a function that accepts the challenge as an input argument, which resembles the notion of a mathematical function.

The response is considered as a unique identifier for the PUF. Thus, its correct acquisition is critical for successful authentication. Unfortunately, this is not a straight-forward process [199]. The first challenge is to measure the unique physical property of the object. Such measurement is unavoidably noisy due to various reasons such as humidity variations, human errors, noise in measurement equipment, small damages to the measured object and so on. The other challenge is how to store the unique physical property. In case of human biometrics, such information has a sensitive aspect, concerning the privacy of a human being. Likewise, encrypted storage of physical object's characteristics is necessary because it may reveal secret structures of the object. However, the noisy measurement process prevents a reliable verification of the newly measured unique physical property with the stored encrypted one.

This unclonable uniqueness can be applied to anti-counterfeiting. If a PUF-based component is embedded into a device, it will make it unclonable. The challenge-response behaviour of PUF significantly changes if it gets damaged or modified, e.g. by an attacker. PUFs can be used in secure key storage in which the response is considered as the secure key [200, 75]. In this application, the response is acquired temporarily by impulsing the PUF module.

Pappu [159] defined PUF with four principles in a challenge-response environment:

1. The object should accept a large set of challenges and provide unpredictable responses.
2. The object should be extremely hard to clone physically.
3. Modelling the challenge-response dynamics should be mathematically difficult.
4. The physical structure of the object must be hard to characterize.

Tulys et al. [199] discuss how each of the above principles can assure one aspect of the physical protection. The first property ensures randomness, the second refers to physical unclonability, the third defines mathematical unclonability and the fourth prevents a potential attacker from replicating the behaviour of PUF by using a simulator. A PUF object based on such definition is highly complex. To satisfy the second principle, the production process must be fundamentally uncontrollable. To satisfy the first and third principles, the response must be sensitive to small modifications to the object's inherent structure. Finally, the principle four is only satisfied when the object is hard to scrutinize.

There have been numerous proposals in academia and industry regarding the applications of PUFs. In [163], the authors propose a tamper evident integrated circuit (IC) with a unique identifier based on active coating. The IC is covered with a layer of conductors and insulators. The sensors inside the IC analyse the coating and generate a binary string. If there is a change in the binary string, it would be interpreted into modifications in the coating. This concept was later developed by Tuyls et al. [201]. They invented the term “coating PUF” and showed how a secure key could be derived from the coating.

Pappu et al. [159, 157] proposed a three-dimensional optical structure with randomly positioned transparent components in 2001. When it is exposed to laser beams, the reflection of the light would result in a random pattern of bright and dark areas, which are called speckles. Speckles have been used in paper sheet identification, too [178]. Sharma et al. [178] measured the generated speckle patterns from a paper sheet surface with a pluggable USB digital microscope. We will explain their approach in Section 3.2 in more detail.

In 2002, Gassend et al. [67] introduced the concept of silicon PUFs. It is based on the fact that during the manufacturing process of an IC, there are inevitable slight differences among circuits. These variations do not harm the operation of the ICs,

while they constitute a source of randomness. In this case, the challenge is a choice of a certain path on an IC, and the response is the delay time of signal transmission through it. This delay is the sum of all delays in the wires and logic gates alongside the path.

Static RAM (SRAM) PUFs were proposed in 2007 [76]. Similar to silicon PUFs, they are based on the random manufacturing variations. SRAM consists of memory cells. They enter into an uncertain state when the SRAM is switched on. Consequently, a cell initiates with either “0” or “1” state, depending on the exact characteristics of the cell. Therefore, a freshly switched-on SRAM is challenged by choosing certain memory addresses and the response would be the returned binary values.

**Types of PUF.** The nature of the challenge-response pair varies in different PUFs. Some PUFs generate a limited number of challenge-response pairs, called as weak PUFs in the literature [170]. In contrast, PUFs with complex challenge-response behaviour are called strong PUFs [170].

Another form of categorization classifies the PUFs based on the object characteristics that are inherent with them. Maes et al. [132] have categorized them as follows:

- *Non-electric PUFs:* These structures are the ones with PUF-like properties but they are not inherently electrical. However, the challenge-response for them is measured by electrical techniques. They are categorized into optical PUFs [159, 157, 182, 203, 94, 69, 204], paper PUFs [15, 53, 32, 33], CD PUFs [84], Radio Frequency(RF)-DNA [61], magnetic PUFs [95, 133], acoustical PUFs [211].
- *Analogue Electronic PUFs:* In this category, the basic measurements of an electric or electronic quality of a PUF is done by analogue methods. These PUFs are as follows: Threshold Voltage ( $V_t$ ) PUF [125], power distribution PUF [88], coating PUF [200, 181] and LC PUF [77].
- *Intrinsic PUFs:* All PUFs in this category are embedded inside integrated circuits. They are divided into two main classes: delay-based intrinsic PUFs and memory-based intrinsic PUFs. The former include arbiter PUFs [120, 117, 68, 136, 171, 154, 155, 91] and ring oscillator PUFs [67, 69, 188] and the later SRAM PUFs [75, 90], butterfly PUFs [75, 115], latch PUFs [186] and flip-flop PUFs [129].

**Challenges.** There are various applications for PUF objects in the literature, including authentication [67, 157], secure key storage [200, 75], key exchange [206, 29],

digital rights management [47, 84] and tamper-protection [85, 200]. The emerging inter-connected devices known as Internet of Things (IoT) create more motivation to integrate these applications into practice [82, 224].

Armknecht et al. [10] categorized the PUF research topics into three classes:

- *Hardware Level*: Discovering or designing new hardware with PUF principles [67, 76, 107, 113, 130, 159].
- *Protocol Design*: Applying PUFs as building blocks for designing new cryptographic protocols [24, 29, 35, 86, 131, 137, 172, 202].
- *Modelling Level*: Investigating the behaviour of PUF and modelling its security [8, 9, 29, 67, 76, 150, 159].

There are various challenges to implementing PUF in real-world applications. Hence, there is more focus on addressing the implementation problems. A PUF is similar to biometrics in many aspects. Likewise, when a PUF response is measured, it is prone to measurement noises. Therefore, it is hard to use current cryptographic primitives for the acquired noisy response. Dedicated measurements, specialized signal processing and quantization techniques could be developed to handle such noises. Moreover, the use of proper error-correction techniques must be considered, so as to reduce the effects as much as possible. However, this introduces another set of problems because external side-information (redundant data) should be stored outside the PUF in an insecure environment. Such exposure might leak information about the PUF itself. This is very similar to biometric in terms of combining it with cryptography; however, there is one vital difference. PUFs, unlike biometrics, have the advantage of freedom in design by nature which would make it possible to increase randomness or create fresh new objects to overcome data leakage [199].

One of the major problems in PUF modelling is the diversity of PUF types. This variety leads to various security models and definitions which might not be applicable generically to other types. Armknecht et al. [10] proposed a super-model for security of PUF, and managed to include different types of PUFs in their modelling.

Furthermore, there have been different evaluation frameworks for PUFs. Maiti et al. [134] collected several evaluation frameworks and proposed a unified scheme that included all aspects of a PUF. It provides a foundation to compare different PUF objects more effectively.

Another current challenge with PUF is the expenses of mass-production [199]. Practical applications of PUFs need to be inexpensive to manufacture. At the same

time, it should be feasible to read PUF responses with conventional readers or low-cost dedicated devices.

On the other hand, PUFs are similar to side-channel attacks in many ways [199]. Therefore, in real-world threat models, they might be prone to engineering attacks that target physical implementations. The security design of a PUF should carefully consider different categories of attackers with diverse skills and resources.

### 2.3.3 Security Printing

Security printing is not a single method. Instead, it is a concept that involves various techniques. Each of them is designed to make counterfeiting, forging or tampering harder for the criminals in a specific way.

Securely implementing the security printing solutions has two important aspects. First, it requires using special materials that are hard to acquire. Second, it requires accurately measured printing. Security printing techniques are generally used to authenticate a genuine product from its counterfeits [119].

**Security Printing Types.** There are two major technologies involved in security printing: covert and overt [119]. Overt technologies are visible to the naked eye. Therefore, the product can be authenticated visually and the verification can be performed by ordinary users. However, the inspection process requires a trained verifier. Moreover, applying these techniques imposes expensive procedures in the supply chain. Some of the more well-known overt technologies are holograms, colour-shifting inks, security threads and watermarks.

In contrast, covert technologies are hidden within the product. They do not need to be verifiable by everyone. Instead, the brand owner is mostly the only one who examines them. Moreover, they are not known to the general public, including the potential attackers. Consequently, these techniques need to be easy to implement and cheap to maintain. The downside of the covert methods is that once they are revealed, preventing the replication becomes difficult. Popular covert technologies include security packaging papers, security inks, UV ink, thermochromic ink, digital watermarks, screen printing, flexographic printing and imprinting and so on.

Banknotes are a perfect example of an item carrying state-of-the-art security printing techniques. Typically, each modern banknote design includes at least 20 types of security features that are not disclosed to the general public [5, p. 441]. Some of these features are known to bank staff or secondary inspectors. However, these features usually leak to the criminals within a few years. Thus, it leads governments to develop new designs every few years.

There are many security printing methods in the industry. However, substrate-based features can be considered as one of the dominant techniques in the field. These features are directly embedded into the product structure during the manufacturing process. For example, special papers are manufactured for banknotes by cotton fibres strongly woven together and randomly scattered Ultra-Violet (UV)-sensitive particles which glitter when exposed to UV emanations.

One of the oldest and most renowned forms of substrate-based security printing is the watermarks. This is a recognizable image or pattern that appears lighter or darker than the surrounding patterns when the object is exposed to light. This type of watermark is sometimes called multi-toner since it varies in tone and can produce complex images with multiple greyness levels.

In another type of substrate-based security solution, special fibres are embedded inside the object during the manufacturing process. They are considered as overt security features and usually, have invisible colours that glitter when exposed to UV light. Security fibres are often utilized in addition to watermarks. Other forms of this category such as magnetic watermarks, which are randomly scattered magnetic fibres, are also used [5, p. 439].

Another group of methods utilize the absence of brighteners in the object's body. For example, banknote paper must be UV-dull which means the paper is free from optical brightening particles. This feature is applied along with other security printing techniques such as fluorescent inks or embedded UV-sensitive fibres.

Security threads are another form of substrate feature. In this method, a metal thread is embedded in the paper or product. Depending on the type of metal, the presence of this thread can be checked with the naked eye, using fluorescent light, optical effects, or machines. When introduced in banknotes, forging this mechanism was thought to be impossible; however, a clever criminal gang came up with a simple way to trick the ordinary users. They printed patterns of the thread on the banknote in a way that it gave the impression of being inside the banknote texture [49].

**Challenges.** Security printing faces some challenges as technology evolves. Although modern technologies would make security printing more accurate and robust, it also gives the criminals more opportunities to attack. The widespread utilization of commodity printers and their improving accuracy and capabilities has resulted in the emergence of a new wave of amateur tamperers [5, p. 436]. This category of attackers are harder to spot for governments since they normally operate on a small scale and the circulation of their products is limited. Overall, the rise of possible

threats motivates security printing solutions to evolve at the same pace as the criminals. As a response, security printing industry shifts to contain digital technologies added to their common security printing techniques [19]. For instance, the new generation of banknotes carries an invisible digital copyright watermark that would help vending machines to detect genuine banknote by processing the digital image taken from it [74]. *Optical document security* [210] refers to the solutions that combine security printing and computer vision algorithms. They aim to deter counterfeiting and tampering by these emerging attackers more effectively.

Meruga et al. [139] proposed a covert security ink based method to print QR codes that are not visible under ambient light. They proposed a combination for security ink that makes the printing more secure against counterfeiting. In their experiments, when beamed with infrared laser, the relevant QR code that is printed with their security ink could be examined for tampering by a mobile phone. Gyrnik et al. [71] discussed that the new trends in security printing are still vulnerable against attacks in many ways. Therefore, they proposed a new system that is combination of analogue document protection methods such as optical holography with digital machine readable ones like computer generated holograms. Then, they created quasi-random machine-readable images. Recording and verification of their special security measurement is done by combined optical and electronic sub-micrometer technology devices. Luciani et al. [126] introduced a multiple-level technique to investigate availability of “Optically Variable Device (OVD)” in an object. They used their method to analyse complete holographic images with various kinetic and dynamic effects. They used positive and negative photo-resist and diffraction gratings in their design. Kiuchi et al. [112] developed an algorithm to analyse security printing lines. They performed a robust frequency-domain analysis on the lines that are printed by different security printing techniques. Other similar research papers include [3, 118, 23].

Finally, other research on proposing different variations of security inks with diverse characteristics and sensitiveness can be found in [221, 218, 80, 114]. These methods are mainly focused on making the manufacturing of the ink harder for an attacker, or making the verification easier for verifiers.

## 2.4 Cyber Tamper-Evident Solutions

The popularity of software products has prompted IT companies to invent various ways to protect them. This protection, however, is not straightforward. First, companies need to combat illegal copying of their product, a.k.a. copy protection mech-

anisms. They also need to make their product packages tamper evident to assure buyers that they are purchasing genuine products. Finally, the growing size and complexity of the software source code, increase the risk of unintentional security flaws. All these threats have motivated IT companies to propose various architectures, protocols and cryptographic solutions to ensure the security, authenticity and integrity of their resources and products.

#### 2.4.1 Source Code Protection

One of the most important challenges for software vendors is to protect their source code from different adversaries. No one can perfectly protect a source code from malicious activities – the attacker needs only one weakness to defeat the whole system. Even, in more sensitive cases, the companies should not trust the compilers for their software development [195]. Moreover, there has been numerous examples in which a source code gets tampered by unknown attackers or even trusted users. Big software including openSSH, IRC, PhPMyAdmin and even Linux Kernel have already been subject to source code tampering.

**Protection against insiders.** In practice, apart from imposing legal obligations on their employees, there is not much a company can do to protect their source code from malicious insiders. Companies tend to rely more on version controlling protocols like Git and SVM than on protecting their source code from their staff. Also, version controlling protocols do not guarantee to keep the source code safe. However, tools for constant static and dynamic analysis of the source code can help. Using hash digest can be another solution to ensure source code integrity. Some physical tamper proof storages devices such as Full Disk Encryption (FDE) drives can reassure the owners about the integrity of their data, too.

**Protection against outsiders.** Protection against software tampering by outsiders mainly relies on two categories of solutions [197]: hardware and software based. In the first category, a secure trusted hardware executes the code. Information technology giants attempted to design trusted hardware platform in recent years. For example, Microsoft heavily invested in a project called Palladium (later changed to Next-Generation Secure Computing Base or NGSCB) to develop a “trusted system” [160]. Microsoft has not released NGSCB officially, yet. However, Boneh et al. [28] demonstrated that such systems are still vulnerable to timing attacks.

In another category, the protection mechanism relies on a software-based solution to ensure the security of the execution environment and the source code. We will focus on the software-based approach in this dissertation.

**Software-based source code protection methods.** Collberg et al. [51] recognized three major attacks on the source code level: Reverse engineering, software piracy and software tampering. They surveyed existing research on defensive mechanisms for each one of these attacks. A defence against reverse engineering is code obfuscation [50, 51]. In this method, the source code is modified in a way that is not readable by attackers, yet the code can still perform its intended tasks. A defence against software piracy is digital watermarking [51] where the origin of the software is embedded inside it secretly. One can verify this watermark to determine the authenticity of the source code. A defence against software tampering is tamper-proofing. This makes the source code sensitive to any unauthorized modification. The source code becomes non-functional when a modification is detected. In this section, we focus on tamper-proof mechanisms.

**Software tampering-proofing mechanisms.** Tamper-proof program module is a part of the software source code that is responsible for detecting or preventing tampering attacks [51]. The challenge to develop and implement any tamper-proof solution is that the part of the program ( $P$ ) which is responsible to handle a tampering situation is also accessible to the attacker, as it is part of  $P$  in the first place.

The detection of tampering in a software program is a challenge, too. There are various scenarios that source code  $P$  should be protected from tampering. For example,  $P$  can be infected with a computer virus that modifies the code.

A tampering-proof program should contain two major parts [51]: First, it should *detect* if the program has been modified. Second, it should *respond* to these modifications accordingly. Ideally, these two phases should be spread randomly over time and space to confuse a potential attacker [51].

**Tampering detection.** Collberg et al. introduced three strategies to detect tampering in software [51]:

- Manual examination of the source code to detect any inconsistency with the original source code. One can compare the hash digests of source codes to make the comparison faster and more efficient.
- Examination of the validity of the output produced by the code. This technique is known as *program checking* [20, 21, 22, 65, 168, 169, 220].
- Generation of the executables on the fly in the hope that minor changes in the code would result into noticeable differences in the final output. This category uses what are called *self-checksumming techniques* [40, 145, 44, 64, 40, 92].

The first approach is usually practised in the more common software products where security and code integrity is not the main concern. In the open-source community, where the source code is publicly available, SHA1 and MD5 digests of all published files are automatically generated and verified by major open-source download repositories such as sourceForge.com [185].

In the second approach, the software architect defines a set of testing suites [220]. A testing suite is a carefully designed collection of inputs to identify bugs and tampering in the performance of the program. It functions based on inputting a test suite to the program, then comparing its output with the expected output.

Selection of a correct testing suite is a challenge though. First, it should be verified to contain a correct set of input/output pairs, second, it should comprehensively cover all the components of the software and third, it should consider different aspects of the program performance such as delay time, resource consumption and so on.

Blum et al. [20] proposed simple-checkers for the testing suite. They considered an input/output pair should be computed in a determined amount of time. Thus, they formalized their method to consider an upper-bound to the output computation delay, too. If output is delayed more than the expected time, it is considered as fail, regardless of the correctness of the output.

The third approach has been first proposed by Aucsmith et al. [64, 11]. They called their method “content protection architecture”. In their method, a program breaks into various partitions and each of them is individually encrypted. The program is executed by decrypting and jumping to partitions based on a sequence that is randomly generated. After execution of each partition, it is re-encrypted. Thus, only the running partition is in plain-text at any time. If that partition is tampered with, then the output of the program would ultimately be different. Furthermore, by tracking the outputs of each partition, one can detect which one is being tampered with.

Another self-checksumming technique is proposed by Chang et al. [40]. This method is called “guarding”. It uses guards to repair the tampered parts when detected. Chen et al [44] proposed a method called “oblivious hashing (OH)”. This approach employs hash-based values to verify the integrity of software. In this method, a certain runtime value is selected and its checksum is calculated and verified.

Self-checksumming approaches are not suitable for all programming languages [51]. Furthermore, it imposes a large execution overhead on the protected program [11].

In recent years, another approach has emerged in the source code tamper evident research called code encryption. In this category, as the name implies, the source

code is encrypted with special cryptographic primitives [36, 37]. Some tamper proof solutions use white-box cryptography in which secure keys are not revealed even when cryptographic operations are being observed in detail by an adversary, i.e. a debugger [208]. Other cryptographic-based tamper protection mechanisms are [108, 38].

**Response to tampering.** In general, two common types of responses are available. The first is to make a program crash right after the detection of tampering and the second is to trap it inside an infinite loop. However, Tab et al. [190] discussed how both approaches have a predictable pattern. They proposed to give a delayed response to tampering. A more recent research trend is tamper-tolerant software. This category reduces the effects of tampering and allows the program to function as it were unmodified [100].

## 2.4.2 Other Aspects

Source code tamper-proof protection is only one aspect of cyber tamper evident mechanisms. In this section, we overview other solutions in this category.

### 2.4.2.1 Copy Protection Mechanisms

Any solution that protects software from illegal reproduction falls into the class of copy protection mechanisms. It helps stakeholders benefit from their product and maintain their copyrights through some kind of digital right management (DRM). Thus, the main duty of such mechanisms is to protect the software against piracy.

Copy protection mechanisms go back to the early days of using tape cassettes as storage media. However, the threat became more serious after digital storage became popular as a container of games, films, audio and software in the 80's and 90's. There have been numerous techniques to overcome software re-copying but none of them were completely efficient.

Initially, the protection approach was hard coded physical bit-level characteristics of the storage media, such as sector structure, that could not easily be replicated by copiers. This category is called "Bit Nibblers" [164]. In the 80's, Apple II's well-known Locksmith protection functioned based on reading the data track by track, ignoring the original sectors that were set in the storage. It was defeated by a postgraduate student shortly after its introduction [144].

Later, the copy protections relied on a proof of purchase from users. The possession of the proof provides *activation* of the software product. The proof of purchase

varies in combination but usually it is the registration information that the buyer already provided to the seller. Microsoft Genuine Advantage is a famous example of such a method.

The proof of purchase based activation has a serious vulnerability. If a genuine user shares his purchase proof to anyone, there is no way to prevent its illegal use. Thus, software vendors try to limit installation by either unique identification of the execution platform or ownership of a special token. It could be the user's MAC address or other hardware-specific information that users can not change. This method is known as dongle-based protection [93].

The rise of virtualisation and web based applications, however, challenged these solutions. The widespread turn to develop web based applications moved the solutions to classic web-based authentication methods such as traditional passwords [176]. Only the users with the correct username and password combination could access the web based application.

Cloud computing involves the latest trend in copy prevention methods. The processes are presented to users through web services. The users should be authenticated while requesting the service and the user's computer is not the platform that runs the service. This gives an opportunity to software vendors to limit their product to only authenticated users and, moreover, to control the execution of it [7].

#### 2.4.2.2 Digital Media Protection

Following the popularity of digital storage of multimedia, the need to protect the author rights and digital forensics urged new emerging techniques to protect the authenticity and integrity of data. Initially, the researchers used hash functions to ensure its integrity; however, a hash digest is not tolerant of noise. Therefore, they coined a method called digital watermarking, enabling the authors to embed some invisible watermarks in digital images.

Friedman [66] proposed an embedded digital signature for each captured image. Their method enables everyone to verify that the image and the issuer are both genuine. Yeung et al. [225] focused on the integrity of an image by using digital watermarks. They used a pseudo-random sequence and a modified error diffusion method to maintain the image integrity. Lin et al. [121] introduced a method to insert authentication related data into the JPEG coefficients. This method makes watermarks more resilient against JPEG compression algorithms and thus, it would make the images more portable. Wong et al. [223] applied a public/private key pair to encrypt grey-scale pictures.

Furthermore, there are different proposals that use digital watermarking to distinguish counterfeiting and tampering. Ho et al. [89] designed an authentication protocol to authenticate holograms on cards by digital watermarking. Their algorithm was based on a transformation coefficient based watermarking. Ono et al. [149] proposed a watermarking scheme to distinguish fabricated 2D barcodes. They used evolutionary algorithms to recognize the secure watermark design. Cheddad et al. [41] proposed a new method to protect digital documents against forgery. They combine 1D hash algorithm with frequency domain features of the digital document for encryption. They considered measures to reduce the noise and effects of image compression algorithms such as JPEG. Guo et al. [79] discussed a new watermarking method based on a technique called “parity-matched error diffusion (PMEDF)”. This technique uses a parity matching strategy to spread the watermark in the pixels of an image. In order to maintain robustness, they applied a voting mechanism to decode the watermark. Their digital watermarking algorithm was robust and resilient against various attacks and modifications.

#### 2.4.2.3 Software Packaging

The ease of cloning software products forced vendors to rely on common packaging solutions to detect counterfeiting, too [5, p. 435]. They had to design their product packaging in a way that any tampering would be evident or prevented. For example, like to other product packagings, software packages are equipped with special seals that reveal malicious tampering. In any case, similar physical tamper-evident protections are applied for this category. We have already explained these techniques in previous sections.

## 2.5 Summary

In this chapter, we reviewed the current approaches in physical and cyber tamper evident technologies. We discussed the security dimensions of tampering and its applications. Then, we analysed three important physical tamper evident solutions: Unique Objects, Physical Unclonable Functions and Security Printing. For each of these solutions, we summarized its current trends, challenges and research. Furthermore, we reviewed cyber tamper evident solutions in the literature. First, we focused on the source code tamper proof solutions and then we explained other trends in cyber tamper evidence.

As discussed, the current state-of-the-art techniques in tamper evidence are not bullet-proof. Thus, in the next chapters we will propose our solutions to develop more advanced tamper evident technologies.

# Chapter 3

## Physical Tamper Evidence: Paper Sheet Case

In this chapter, we will propose a cost-effective method to derive unique patterns of a paper sheet, aka “*paper fingerprint*”<sup>1</sup>. We utilize an off-the-shelf digital camera with macro capturing capabilities and an ordinary light source. We evaluate our method across a large number of paper sheet samples. We recognize each paper sheet based on its fingerprint with perfect accuracy (100% success rate in around 500,000 comparisons).

The captured paper fingerprints carry a relatively large entropy. Our analysis demonstrates a paper fingerprint have 809 bits (out of 2048 bits) entropy. Moreover, we examined the effects of different cases of rough handling in both environmental changes and distortion to the paper texture. Our method can successfully withstand these circumstances, too.

Furthermore, we will discuss various security aspects of our paper fingerprinting method. We will argue different approaches that could be applied to securely authenticate a paper sheet. Then, we will propose an authentication protocol based on Hao et al. [87]. Their proposed mechanism is designed for iris recognition; however, their authentication protocol can be applied to paper fingerprinting due to the similarities between paper and iris textures.

Paper fingerprinting can be applied to either tamper evident or anti-counterfeiting solutions. However, we argue that any real-world adaption of “paper fingerprinting” depends on its specific application. Nevertheless, the ability to *track* a paper sheet would prevent its “misuse”, eventually leading to provide tamper evidence. In addition, secure authentication of the fingerprint could prevent counterfeiting by determining the authenticity of a paper sheet. Our proposal would consider both aspects;

---

<sup>1</sup>The content of this chapter has been published in [196]

however, the context of a paper sheet is considered out-of-scope in our research. The cryptographic binding of a paper sheet to its contextual data could potentially be a separate research topic for the future.

### 3.1 Introduction

**Secure paper documents.** Designing secure documents that provide high levels of security against physical forgery is a long-standing problem. Even in today’s digital age, this problem remains important as physical paper is still prevalently used in our daily lives as a means to prove data authenticity, for example, in receipts, contracts, certificates, and passports. A recent trend in this area (e.g., in e-passports) is to embed electronics such as RFID chips within the physical document in question [102]. However, the security of such solutions depends on the tamper-resistance of the chip which must securely store a long-term secret [45]. This tamper-resistance requirement can significantly increase the cost of production. In view of the importance of ensuring the authenticity of paper documents, researchers have been exploring applying digital technologies to prevent counterfeiting. One promising method is based on measuring the unique physical properties of paper that are impossible to clone.

**Paper fingerprinting.** Manufacturing a paper sheet is a complex process and each paper sheet is a unique product from that process. Typically, wooden particles are used as the base, and multiple substances are subsequently applied to stick these particles together to stabilize their placement and shape a thin, usually white, steady surface which we call paper.

In an article published in *Nature* in 2005, Buchanan et al. observed that the surface of a paper sheet is imperfect – it contains random non-evenness as a natural outcome of the paper manufacturing process [31]. They propose to utilize the surface imperfections to uniquely identify the paper. Their method is to use a focused laser beam to scan a pre-designated region on the paper sheet from four different angles, and continuously record the intensity of the reflected laser. The recordings then constitute a unique digital representation of the paper, called “paper fingerprint”. Therefore, Buchanan et al.’s method [31] is the basis of a number of follow-up works, notably [205, 175].

Clarkson et al. (IEEE S&P, 2009) subsequently showed that a commodity scanner could be used to effectively extract paper fingerprints based on the same surface imperfections [48]. Their method is to scan the paper surface from four different angles and then construct a 3-D model. Then the 3-D model is condensed into a concise feature vector, which forms the paper fingerprint.

Later, Sharma et al. (CCS, 2011) proposed another approach named PaperSpeckle, which uses a microscope with a built-in LED as the light source to extract the paper speckle patterns at the microscopic level (1–2 microns) [178]. The underlying idea

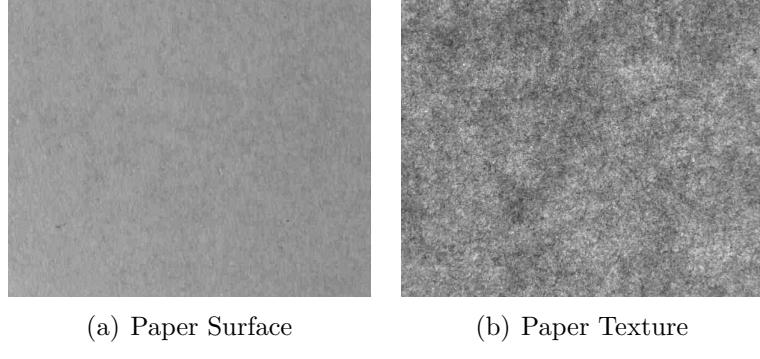


Figure 3.1: The surface and texture of the same area of a paper sheet as captured by a camera based on a) reflective and b) transmissive light.

in PaperSpeckle is based on the concept of speckles: i.e., when light falls on a paper sheet, the scattered light forms randomly mixed bright and dark regions, which can then be captured by a microscope. The captured image can be further processed to produce a compact binary fingerprint.

**Our idea.** So far, prominent works in this area have primarily focused on the imperfections of the paper surface. In contrast, our work is inspired by the observation that the wooden particles constituting the building blocks of a paper sheet scatter over the paper quite irregularly. We hypothesize that this irregular placement of wooden particles provides a unique pattern, which can be extracted and used as a paper fingerprint. We call the unique pattern caused by the random interleaving of wooden particles the *texture* of paper.

Unlike previous works that measure the paper surface characteristics, we propose to fingerprint a paper sheet based on measuring the paper texture patterns. We capture the texture by putting a light source on one side of the paper and using a commodity camera to take a photograph on the other side. This is intuitively based on the common observation that putting a paper sheet under light will immediately reveal rich irregular textural patterns visible even to the naked eye. Figure 3.1 shows the difference between photos taken of the paper surface (based on reflective light) and of the paper texture (based on transmissive light).

## 3.2 Paper Texture

When light falls on an object, it is partly absorbed, partly reflected, and partly transmitted, and paper is no exception. Absorption occurs based on the resonance

principle: the energy of the light waves of a specific frequency is absorbed and transformed into kinetic energy by electrons of the same frequency. The part that is not absorbed, is either reflected or transmitted depending on how opaque (or conversely transparent) the paper is.

Different types of paper behave differently in terms of how much light they absorb, reflect or transmit. This behaviour depends, among other factors, on pulp material, density, thickness and coating substances. Opacity, as defined by the ISO 2471 standard [98], can be seen as an indicator of how much light is impeded from transmitting through the paper, with the opacity of 100% defined for fully opaque papers. Typical office printing paper, with grammage between 75 to 105  $g/m^2$ , has opacity between 86% to 94%. To put this in perspective, opacity for newsprint paper (typical grammage: 40–49  $g/m^2$ ) is in the range 90–94% and for tracing paper (typical grammage: 60–110  $g/m^2$ ) is in the range 24–40% [72]. These values suggest that a considerable proportion of light transmits through such paper, which forms the basis of our proposal to fingerprint paper based on its textural patterns.

Intuitively, the textural patterns created and stabilized throughout the paper in the process of manufacturing can provide a promising source for paper fingerprinting. These patterns are naturally occurring and appear random. Moreover, they are embedded within the bonded structure of the paper and hence are relatively well-protected against manual handling of paper. They are generated as a result of the wooden particles randomly interleaved during the manufacturing process. Finally, once in the finished product, the randomly interleaved wooden particles can not be altered without damaging the paper, hence making any tampering act evident.

To capture the embedded textural patterns of paper and subsequently extract a fingerprint, we limit ourselves to a single photo taken by a commodity camera. This makes our solution more practical and quicker than the previous proposal [48] that has to take multiple scans (on paper surface) from four different angles in order to compute a fingerprint. We note that a single photo is feasible in our case because the paper texture contains richer features than the paper surface, such as the thickness of the overlaying wooden particles, randomly distributed impurities, and different embedded materials with varying opacities. In the rest of the paper, we conduct experiments to show that we can reliably extract a paper fingerprint from the textural patterns.

**Applications.** A vast number of official and legal documents, certificates, official receipts and invoices are printed on regular office paper (sometimes with watermarks, holograms or other security measures), thermal paper, or other types of paper. A property that the majority of these types of paper have in common is that they are not

completely opaque. This means that a considerable amount of light passes through them. Furthermore, embedded irregular textural patterns as a natural result of the manufacturing process seem to be a universal property of all these different types of paper. Consequently, there is considerable potential for exploiting paper fingerprints extracted from embedded textural patterns in order to validate the authenticity of such official and legal documents.

### 3.3 Related work

In the introduction (Section 3.1), we highlight three prominent works in the field (Nature 2005, IEEE S&P 2009, CCS 2011), which have inspired our work. In this section, we conduct a more comprehensive review of the related work.

**Special paper.** Some researchers proposed to fingerprint paper by embedding special materials. Bauder [16] was the first to propose the idea of a certificate of authenticity (COA), which is a collection of fibers randomly positioned in an object and permanently fixed by using transparent gluing material. Once an end-point of a fiber is exposed to light, the other end is illuminated, and as a whole this creates unique illuminated patterns, which can be captured by a light detector. The main intended application is to use COA for banknotes to ensure authenticity. Kirovski [110] followed up Bauder’s work and proposed to combine the captured illuminated patterns with arbitrary text, signed with the private key of the banknote issuer. The signature is then encoded as a barcode and printed on the banknote. Chen et al. [43] proposed an improved scanner to achieve automated verification of fiber-based COAs. In a similar work, Bulens et al. [34] proposed to embed different material—ultra-violet fibers—into the paper mixture, and use a UV scanner to obtain a unique fingerprint. The authors report that the derived fingerprints have 72-bit entropy.

**Unmodified Paper and using laser.** One limitation with all the works mentioned above is that they require modifying the paper manufacturing process. Other researchers investigate fingerprinting techniques that can work with ordinary paper without altering the manufacturing process at all. One prominent work along this line of research is due to Buchanan et al. [31] published in Nature 2005. The researchers proposed to use a focused laser beam to scan across a sheet of standard white paper and continuously record the reflected intensity from four different angles by using four photodetectors. The laser reflection is random, and is determined by the non-uniform paper surface. Hence, the recorded reflection intensities constitute a unique fingerprint. Beijnum et al. followed up this research idea in [205]. They

formulated a criterion for recognition that limits the false acceptance rate (FAR) to 0.1%. Samul et al. [175] presented a similar idea of shooting a beam of laser onto the surface of the paper. But instead of using photodetectors, they proposed to use a CCD camera to capture the microscopic patterns of speckles.

**Unmodified paper and using light.** Laser-based fingerprinting methods have the limitation that they require special laser equipment. A more cost-effective solution is to use a commonly available light source. Metois et al. [141] proposed custom-built equipment called the “imager”, which consists of a consumer-grade video module and lens, housed along with an embedded lighting apparatus. The imager provides a grayscale snapshot of the naturally occurring inhomogeneities of the paper surface. The snapshot is then processed into a vector of real numbers. The authentication of paper fingerprints is based on computing the correlation coefficient between vectors. The equal error rate (EER) is reported to be about 9%.

Clarkson et al. [48] proposed a similar method to fingerprint a paper document by using a commodity scanner instead of a specially built “imager”. Their work was motivated by the observation that when viewed up close, the surface of a sheet of paper is a tangled mat of wood fibers with a rich three-dimensional texture that is random and hard to reproduce. Utilizing the embedded light emission, the researchers use a commodity scanner to scan a paper sheet in four different orientations. Then a 3-D model is constructed based on these four scans. Furthermore, the 3-D model is compressed into a feature vector through computing Voronoi distributions in the scanned region. The comparison between two feature vectors is based on computing the correlation coefficient.

Pham et al. [161] adopted the same approach as Clarkson et al. [48] by using an EPSON 10000XL scanner at 600 dpi to collect 10 scans of the paper surface. In particular, they look at the case when text has been printed over the authentication zone, and propose two methods of pixel inpainting to remove printed text (or marks) from the authentication zone in order to allow ordinary correlation to be performed. Different from the proposed method of Clarkson et al. [48] that compresses the scanned images into a compact feature vector and compares feature vectors based on the correlation coefficient, Pham et al. proposed to use alpha-masked imaging matching to compare regions of the two paper surface images. Improvements are demonstrated using the collected data sets in their experiments.

Sharma et al. [178] proposed a different surface-based fingerprinting method. Unlike prior paper fingerprinting techniques [48, 161] that extract fingerprints based on

the fiber structure of paper, their method uses a USB microscope to capture the “surface speckle pattern”, a random bright and dark region formation at the microscopic level when light falls on the paper surface. The captured patterns are then processed into a vector of digits, which form the unique fingerprint. Fingerprints are compared based on the Euclidean distance between the two vectors.

Beekhof et al. [17] proposed a fingerprinting method based on measuring random micro-structures of the paper surface. The random micro-structural patterns are captured by using a mobile phone camera with macro lens mounted. The captured image is compressed into a binary feature vector by first hashing the image values into a list of codewords and then running the decoding process through a *reference list decoding* (RLD) technique. Two feature vectors are compared based on the Hamming distance.

Smith et al. [184] proposed another method to capture light reflections from paper surface. Their method involves printing an 8mm box on paper, and then taking a snapshot of it. The alignment can then be done automatically by software based on the printed box, but no details are given in the paper. The authors apply a “texture hash function” to generate the fingerprints. Fingerprints are authenticated based on computing correlations of the texture hash strings.

Haist and Tiziani [81] proposed a method to fingerprint German banknotes by using a CCD digital camera to take a snapshot of a banknote based on transmissive light. Then, the snapshot is saved as a JPEG image (2.86 KB), which, along with a digital signature, is printed on the banknote as a string of 3250 ASCII characters on an area of  $5 \text{ cm}^2$ . However, no prototype implementation is reported. The verification is performed by applying the Fourier transform to obtain a feature vector and computing the correlation between the two vectors. Their idea is the closest to ours in terms of using transmissive light. However, our work is substantially different from theirs in several important aspects. First, their work involves testing only three German Deutsche Mark banknotes while the test data sets used in our work are far more extensive. Second, they do not perform image pre-processing. As a result, the positioning and orientation are done manually rather than automatically by a software algorithm as in our case. Third, they do not carry out image encoding. Consequently, they need to store a JPEG image (2.86 KB), while we only need to store a compact fingerprint (256 byte). Fourth, they do not implement their idea in a prototype system. Hence the feasibility remains uncertain. Most importantly, the Haist-Tiziani paper does not report any error rate performance, or any entropy analysis, and it does not perform extensive robustness tests as we have done.

Renesse [209] proposed a 3-dimensional-structure authentication system (3DAS) to authenticate a standard PVC ID-card that has a  $3 \times 3$  mm<sup>2</sup> 3DAS-structure in a transparent window. The 3DAS-structure contains spunlaid fibers that are thermally bonded at their cross points. In their experiment setup, two infrared emitting diodes (IREDs) are used as lighting sources to shine on the 3DAS area from two different angles. This creates two shadow images that are then captured by a two-dimensional CCD-array. By alternatively switching both IREDs the required parallel images are produced. Finally, a 20-byte fingerprint is obtained by calculating the centres of gravity of the captured images. However, Renesse's paper does not report error rate performance or perform any entropy analysis. It does not report robustness tests either.

**Summary.** We have presented related paper-fingerprinting techniques proposed in the literature, which have different requirements on paper material, use different types of illuminating sources and scanning equipment, apply different signal processing techniques and obtain fingerprints of different types and features. Our work advances the state-of-the-art in this field by presenting the first practical solution that works with ordinary paper, uses an ordinary lighting source combined with an off-the-shelf camera, takes only 1.3 seconds to produce a compact fingerprint (256 bytes) from one snapshot, achieves an ideal 0% FRR, 0% FAR as well as very high entropy (807 bits) in fingerprints, and is demonstrably robust against rotation, crumpling, scribbling, soaking and heating. The near perfect result is attributed to the idea of capturing the paper textural patterns through transmissive light. As detailed in Section 3.5.2, using transmissive light reveals richer textural patterns than reflective light and produces more reliable features. This explains our superior result as compared to the earlier surface-based paper fingerprinting methods [31, 48, 178].

### 3.4 Texture to the Rescue: Practical Paper Fingerprinting based on Texture Patterns

In this section we discuss a high level description of our proposed method for capturing paper textural patterns and extracting a reliable and unique paper fingerprint from those patterns. To be able to capture paper textural patterns, we take a digital photograph of the paper sheet through which light is projected. Then, we need to perform a series of preparation operations such as aligning and resizing of the original image. Afterwards, in the texture analysis phase, we utilize 2-D Gabor filter [59] to extract textural information from the captured image. Subsequently, we propose a

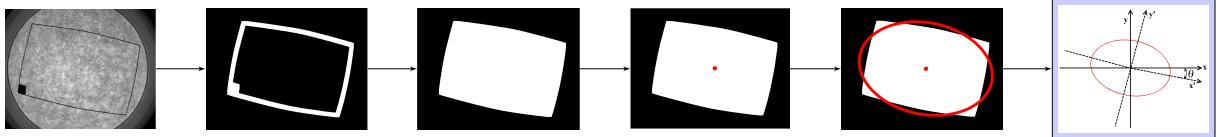


Figure 3.2: Step-by-step rotation recognition process in the preparation phase.

simple paper fingerprint extraction method that generates a binary string, the paper fingerprint. Once paper fingerprints are in the binary string format, they can be compared using well-known methods, such as computing the fractional Hamming distance between any two paper fingerprints. In the following, we give more details about the preparation phase, Gabor transform, the fingerprint generation method, and the fingerprint comparison method based on fractional Hamming distance. Further implementation details and settings of our experiments will be discussed in Section 4.6.

### 3.4.1 Preparation Phase

The preparation phase consists of operations of identifying the designated area of the photo which is to be used for fingerprint extraction and aligning the image in terms of movement and rotation. To indicate the fingerprinting area, we print a small rectangular box on the paper sheet. In addition, we print a filled square on the bottom left of the box, to allow automatic alignment by our implementation.

As shown in Figure 3.2, aligning the rotation of the image involves several steps. First, we start with a photo of the fingerprinting area. The photo is converted into grey scale. The printed region (the rectangular box and the filled square) can be easily identified by applying a grey-scale threshold. This threshold is computed by the Otsu method [151], which chooses the threshold in a such way to minimize the interclass variance of black and white pixels. We have applied the same approach for both reflection and transmission analyses. We observe that the borders in both reflection and transmission samples were recognized correctly using this technique. The result is a binary image: “0” for black and “1” for white. This simple thresholding may also produce some “noise” scattered around the image, but they can be easily removed based on area. To ensure the borders of the printed rectangle are connected, we draw a convex hull of the outer pixels to form a connected shape.

Once the printed rectangle is identified, we fill up the region within the rectangular border with the binary value ‘1’ (white). We identify the centre of mass of the rectangular object based on computing the first-order moment [191] and use that as

the new origin of the Cartesian coordinate system. This corrects any misalignment due to paper movement.

Then, we need to correct any misalignment caused by rotation. This is based on computing second-order moments [191] in the new Cartesian coordinate system. Let  $B(x, y)$  denote the binary 2D object in Cartesian coordinates representing the recognized rectangular box area. There are three second-order moments as follows:

$$\begin{aligned} u_{20} &= \iint x^2 B(x, y) dx dy \\ u_{11} &= \iint xy B(x, y) dx dy \\ u_{02} &= \iint y^2 B(x, y) dx dy \end{aligned}$$

The rotation of the binary 2D object  $B(x, y)$  can now be calculated as follows:

$$\theta = \frac{1}{2} \tan^{-1} \left( \frac{2u_{11}}{(u_{20} - u_{02}) + \sqrt{(u_{02} - u_{20})^2 + 4u_{11}^2}} \right) \quad (3.1)$$

The above formula – based on a method originally proposed by Teague [191] – calculates the angle between the  $x$  axis and the major axis of an ellipse that has equal second moments to the recognized rectangular box. It gives us the counter-clockwise rotation of the object with respect to the horizon. After  $\theta$  is calculated, the image can be rotated accordingly.

Three-Dimensional rotation of a captured texture is intentionally discarded because of two reasons. Firstly, the current position of camera and the light source makes it difficult for a regular user to deliberately take a picture that needs 3-D rotation. Moreover, such a photo would inevitably disturb the existing ratio of the rectangular edges; thus it would need more adjustments than 3-D rotation to make the paper texture fit for fingerprinting.

It is worth noting that in the captured image, the borders of the rectangles are slightly curved rather than being straight due to lens artefact. This slight curvature does not affect our alignment algorithm. We use the raw bitmap image acquired from the camera instead of the processed jpeg image. This raw image is stored separately in the camera in the “.rw2” format and contains the raw information captured by the camera sensor without any processing.

After rotation is corrected, the image is delimited to the lowest and highest  $x$  and  $y$  values of the coordinates of the pixels inside the recognized rectangular box. This image is denoted by  $I(x, y)$ . Since the borders of the rectangular box are slightly curved,

$I(x, y)$  includes tiny areas around the four edges that are outside the box. Those, along with the printed rectangle and the filled square, are identified in a binary mask vector (similar to the identification of eyelids and eyelashes in iris recognition [57]) and will be excluded later in the Hamming distance comparison process.

### 3.4.2 Gabor Filter

Gabor filters are mainly used for edge detection in image processing. Besides, they have been found to perform efficiently in texture discrimination. Gabor filters are able to extract both coherent and incoherent characteristics of textural patterns [55]. Coherent properties are the patterns which remain unchanged between snapshots of the same sample while incoherent ones refer to the patterns which change between snapshots of different samples. The two dimensional Gabor wavelets are popular in biometric recognition problems such as iris recognition [57], fingerprint recognition [116] and face recognition [123]. A Gabor filter's impulse response is basically that of a Gaussian filter modulated by a sinusoidal wave. Consequently, Gabor filters capture features in both the frequency and spatial domains. Generally speaking, a Gabor filter would consider the frequency of a pattern (“what”) as well as the two-dimensional (2D) position of the pattern (“where”) [55]. Let  $\exp$  be the natural exponential function. The 2D Gabor wavelet is calculated as follows using Cartesian coordinates:

$$G(x, y) = \frac{f^2}{\pi\eta\gamma} \cdot \exp\left(\frac{\eta^2x'^2 + \gamma^2y'^2}{2\sigma^2}\right) \cdot \exp(2\pi ifx') \quad (3.2)$$

for  $x' = x \cos(\theta) + y \sin(\theta)$  and  $y' = -x \sin(\theta) + y \cos(\theta)$

where  $f$  is the frequency of the sinusoidal wave,  $\eta$  and  $\gamma$  are constant factors that together determine the spatial ellipticity of the Gabor wavelet,  $\theta$  represents the orientation of the ellipticity, and  $\sigma$  is the standard deviation of the Gaussian envelope.

Depending on the frequency of the sinusoidal wave and the orientation of their ellipticity, Gabor filters are capable of discriminating different textural characteristics. Usually, Gabor filters with a range of different frequencies, known as *scales*, and a range of different orientations are applied to find out the best combination of scale and orientation for a specific texture analysis problem. For a fixed maximum frequency  $f_{\max}$  and a maximum of  $U$  scales, each scale index  $u$  defines the frequency  $f$  used in Equation 3.2 as follows:

$$\forall u \in \{1, 2, \dots, U\} : f = \frac{f_{\max}}{\sqrt{2^{u-1}}} \quad (3.3)$$

For a maximum of  $V$  orientations, we consider  $V$  angles equally distributed from 0 to  $\pi$ . Each orientation index  $v$  defines the orientation  $\theta$  used in Equation 3.2 as follows:

$$\forall v \in \{1, 2, \dots, V\} : \theta = \frac{v - 1}{V} \pi \quad (3.4)$$

We apply a Gabor filter to grey-scale images. Let  $I(x, y)$  represent the grey-scale image using Cartesian coordinates. The result of the application of Gabor filter  $G(x, y)$  is simply the 2D convolution of  $I$  and  $G$  as follows:

$$C(x, y) = I(x, y) * G(x, y) = \iint I(x, y) G(x - \eta, y - \xi) d\eta d\xi$$

The result  $C(x, y)$  is a complex number for each  $x$  and  $y$ .  $C(x, y)$  can be alternatively viewed as a matrix with the discrete values of  $x$  and  $y$  mapped to the columns and rows. Throughout the paper, we use functions defined over Cartesian coordinates and matrices interchangeably.

### 3.4.3 Fingerprint Generation

Our fingerprint generation method takes the output of a Gabor filter and produces a binary string. Let the element located in row  $j$  and column  $k$  of the matrix  $C(x, y)$  be  $m_{jk} = a + bi$ . We define a 2-bit Gray code based on which quarter of the complex plane the element  $m_{jk} = a + bi$  falls in (see Figure 3.3). For example, when  $a$  and  $b$  are both positive, the encoded value will be 11. Thus, every element in the matrix is replaced by two bits. The result is a binary string which we call the paper fingerprint.

### 3.4.4 Fractional Hamming Distance

After paper fingerprints are generated, fractional Hamming distance between any two fingerprints can be used to compare them. Hamming distance is simply the number of positions in which the bits disagree between two fingerprints. This is a classical bit error rate (BER) metric in communication. Fractional Hamming distance is the normalized version, resulting a value between 0 and 1. Usually masking is used to discard the effect of irrelevant bits in a fingerprint. For each fingerprint, a mask is defined as a binary string of the same length in which bits corresponding irrelevant positions are set to 0 and bits corresponding effective positions are set to 1. The masks are calculated in the preparation phase as discussed above. Given two fingerprints  $f_1$  and  $f_2$ , and their corresponding masks  $m_1$  and  $m_2$ , the fractional Hamming distance is calculated as follows:

$$HD(f_1, f_2, m_1, m_2) = \frac{\|(f_1 \oplus f_2) \cap m_1 \cap m_2\|}{\|m_1 \cap m_2\|} \quad (3.5)$$

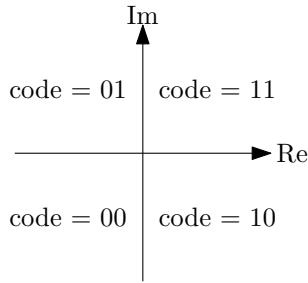


Figure 3.3: Gray code for a complex value  $m_{ij} = a + bi$  in the complex plain.

where  $\oplus$  denotes the bitwise exclusive-OR (XOR) operation and  $\cap$  denotes the bitwise AND operation. A relatively small fractional Hamming distance indicates that the two fingerprints are likely to belong to the same paper sheet, while a relatively large fractional Hamming distance (around 0.5) indicates that the two fingerprints are likely to belong to different paper sheets. In the rest of the paper, we simply use Hamming distance (or HD for short) to refer to fractional Hamming distance.

## 3.5 Evaluation

In order to evaluate our suggested method for paper fingerprinting, we collected several datasets in different situations. In this section, we first explain the parameter settings and experiment configurations under which we carried out our evaluations. Then, we provide the details of the evaluation framework that we use to assess the results of our experiments. In particular, we consider metrics used for evaluating the effectiveness of biometric systems as well as those used for evaluating the effectiveness of physical unclonable functions (PUFs), since paper fingerprints can be seen as both. Subsequently, we give results that justify the choices we had to make in terms of how we collect our datasets and settings we use for Gabor filter. Finally, we give the details of our main dataset collection and provide the results of our experiments, including evaluation of the proposed method against biometric and PUF metrics.

### 3.5.1 Parameter Settings & Experiment Configurations

In order to obtain consistent fingerprints, we require that a relatively small but fixed part of a sheet of paper is used as a source of fingerprint extraction. We chose to print a rectangular box (37mm×57mm) on the sheet to indicate this area. In addition, we printed a small filled square (5mm×5mm) at the bottom left of the box (see Figure 3.10). Using this small square, in our preparation phase our method can

check that the rotation has been carried out correctly (distinguishing cases when the paper is placed upside-down or flipped).

The original photos in our experiments are all  $3456 \times 4608$  pixels. After the preparation phase, we get a corrected and delimited image of variable size, ranging between around  $2300 \times 3300$  pixels to  $2350 \times 3350$  pixels. This image is then resized to a  $640 \times 640$  pixel image  $I$  which is then given as input to Gabor filter. The rectangular size conversion is for the convenience of applying Gabor wavelets in the next stage to produce 2048 bits in the output (the same size as an iris code). We use a Gabor impulse response of size  $100 \times 100$  and the output of Gabor filter in our experiments,  $C$ , is a complex matrix of size  $640 \times 640$ . This matrix is downsampled to one of size  $32 \times 32$ , before being given as input to the fingerprint generation algorithm. This downsampling process is done by simply picking the elements in every 20th row and 20th column. Fingerprint generation replaces each complex value with two bits. Hence, the final paper fingerprint is a string of the size  $2 \times 32 \times 32 = 2048$  bits.

We chose to downsample the output of the Gabor filter for two reasons. First, it makes the data storage more compact. With 2048 bits (256 bytes), we are able to store the fingerprint in a QR code as part of an authentication protocol (we will explain the protocol in more detail in Section 3.7). Second, adjacent pixels in the image are usually highly correlated. Hence, downsampling serves to break the correlation between bits. Through experiments, we found this simple downsampling technique was effective to produce reliable and unique fingerprints.

All images have been captured by a Panasonic DMC-FZ72 camera with a resolution of 16.1 Mega-pixels. We chose this camera for two main reasons: the ability to capture a photo in macro mode from a short distance (minimum 1 cm focus) and the ability to mount a macro flash ring over the lens. However, these characteristics are not unique to this specific camera and many other cameras available in the market provide the same characteristics. We mounted an off-the-shelf common macro flash ring on the camera lens, to maintain a constant distance between the lens and the paper surface where the texture is photographed. The camera and its accessories are shown in Figure 3.4(a). In our experiments, we do not use the flash of the macro flash ring; the light source is an ordinary office overhead projector as shown in Figure 3.4(b). The light that the overhead projector provides is intense and adjustable. Furthermore, it has a flat surface with constant distance from the light source. This allows us to put the paper on the surface and then the macro ring resting on top of it before the camera takes a photo of the paper texture. The use of the macro ring also serves to shield the effects of other ambient light sources (e.g., daylight, office



(a) Camera and macro flash ring.  
 (b) Overhead projector as light source.  
 (c) Light box (tracing pad) as light source.

Figure 3.4: The equipment used in our experiments.

lighting). In Section 3.6.3, we will explain the effect of the light source by using an alternative source: a commodity light box (tracing pad) as shown in Figure 3.4(c).

Our evaluations were performed on a PC with an Intel Core i7-2600S CPU @ 2.80 GHz with 8 GB of memory. The operating system was 64-bit Windows 7 Enterprise and we used Matlab R2015a (64-bit) to develop our algorithms.

### 3.5.2 Evaluation Framework

Our work is closely related to the fields of biometrics and Physical Unclonable Functions (PUFs). Biometrics is the science of authenticating humans by measuring their unique characteristics and have a long history of research. A paper fingerprint works similar to biometrics, except that it measures unique characteristics of a physical object instead of a human being. Hence, common metrics that measure the error rate performance of a biometric system apply to our work too. On the other hand, paper fingerprints are related to Physical Unclonable Functions, which is a relatively new field starting from Pappu et al.’s seminal paper published in *Science* in 2002 [156]. Typically PUFs require a challenge and response dynamic, but according to the definition by Maes in [127], paper can be regarded as a “non-intrinsic” PUF, i.e., a PUF that does not contain the circuitry to produce the response on its own. Hence, the same evaluation methods in PUF are also applicable to paper fingerprints.

Because of the close relation to these two fields and their respective evaluation frameworks, we evaluate our method based on metrics used in both fields for a comprehensive analysis. In the following we give a brief description of these metrics. We

discuss Hamming distance distributions, decidability, and recognition rates including false rejection and false acceptance rates in the former category of metrics. In the latter category, we consider uniformity and randomness in the space dimension, reliability and steadiness in the time dimension, and uniqueness and bit aliasing in the device dimension.

**1) Biometric Metrics:** A biometric authentication problem is a specific case of a statistical decision problem in which one decides if two given biometric measurements belong to the same source or not. In order to provide necessary information about the effectiveness of such a biometric, the parameters of the so-called biometric decision landscape need to be specified [56].

If Hamming distance is used for comparison, as it is in our case, the distributions of Hamming distance for two groups of comparisons need to be determined: for comparisons between paper fingerprints from the different samples originating from the *same* paper sheet, and for comparisons between paper fingerprints originating from *different* paper sheets. These are called *same-group* and *different-group* distributions, respectively. The terms *same-group* and *different-group* are the label for a Hamming distance comparison and does not imply dependence between the extracted fingerprints.

For an effective biometric, the same-group and different-group distributions should be well-separated. This makes the decision problem solvable. Let  $\mu_1$  and  $\mu_2$  denote the means, and  $\sigma_1$  and  $\sigma_2$  the standard deviations of the two distributions. Daugman defines the *decidability* metric  $d'$  as follows [58]:

$$d' = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}} \quad (3.6)$$

where  $|\cdot|$  denotes absolute value. Decidability as defined above is indicative of how well-separated the two distributions are: the further and the more concentrated the distributions are, the higher will the decidability be. To give an idea about typical values, the decidability of iris recognition, a well-established and effective biometric method, is  $d' \approx 14$  in an ideal measurement environment and  $d' \approx 7$  in a non-ideal environment [58].

After determining the same-group and different-group distributions, one decides a threshold value situated between the two distributions. Subsequently, the decision on whether two reported biometrics belong to the same origin or not is then made by computing the Hamming distance between the two biometric samples and comparing it to the threshold. For an effective biometric, measurements from the same origin

have relatively low Hamming distance and hence fall below the threshold, whereas measurements from different origins have relatively high Hamming distance and fall above the threshold. If the distributions are completely separated, the decision is correct all the time. However in practice usually there is some overlap between the two distributions. The proportion of biometrics from different origins falsely accepted as being from the same origin is known as the *false acceptance rate (FAR)*. The proportion of biometrics from the same origin falsely rejected as being from different origins is known as the *false rejection rate (FRR)*. For an effective biometric FAR and FRR should be low – ideally zero.

A widely used measure of effectiveness of a biometric is *degrees of freedom (DoF)*. DoF is a measure of the combinatorial complexity of the biometric test, or in other words the number of bits in a biometric measurement that are independent [58]. Consider a biometric that provides degrees of freedom  $N$ , that is,  $N$  independent and unpredictable bits. A comparison between two such biometrics from different origins can be modelled as the probability that a threshold number of  $N$  independently chosen bits agree. Hence, the different-group distribution for such a biometric would follow the binomial distribution with mean  $\mu = p$  and variance  $\sigma^2 = Np(1 - p)$ , where  $p$  is the probability of single bit agreement. Hence, the degrees of freedom for a biometric with a different-group distribution that follows a binomial distribution with mean  $\mu$  and variance  $\sigma^2$  can be calculated as follows:

$$N = \frac{\mu(1 - \mu)}{\sigma^2} \quad (3.7)$$

**2) PUF Metrics.** Paper fingerprinting can be seen as an optical physical unclonable function (PUF) [156], as pointed out in the literature [127, 128]. However, previous works on paper fingerprinting did not evaluate their results in this context. We believe that evaluating our results against established PUF metrics provides further information about the effectiveness of our method and helps put our results in perspective within PUF literature.

We follow the unified framework put forward by Maiti et al. [135]. This framework provides metrics to evaluate a PUF in three dimensions: space, time, and device. In our case, PUFs are the paper fingerprints, and devices are the different paper sheets. Each of these dimensions quantifies a specific quality of a fingerprint: the space dimension analyses the overall variations of fingerprints, the time dimension indicates same-group consistency, and the device dimension discusses the different-group diversity of fingerprints.

Before describing these dimensions, let us define the symbols we use in this framework. Here we consider *effective fingerprints*, denoted by  $r$ . The effective fingerprint is the result of applying the appropriate mask over the original fingerprint  $f$ . We use the following parameters:  $L$  is the number of bits in each fingerprint (2048 in our setting).  $T$  refers to the number of samples taken from each paper sheet in a dataset (e.g., in the our benchmark dataset  $T = 10$ ).  $N$  is the total number of paper sheets involved in a dataset (e.g., in the our benchmark dataset  $N = 100$ ). We use the following indices accordingly:  $n$  denotes the paper sheet number within different sheets,  $t$  represents the sample number within the samples from the same paper sheet, and  $l$  shows  $l$ -th bit in the effective fingerprint.

### 3.5.2.1 Space Dimension

This dimension is concerned with bit variations with respect to the locations of the bits in fingerprints. Metrics in this dimension evaluate the overall inter-sheet behaviour of fingerprints.

- *Uniformity*: This metric shows how uniform 0s and 1s are in a fingerprint. The ideal value for this metric is 0.5. Uniformity of the fingerprint from the  $t$ -th sample and  $n$ -th sheet is calculated as follows [135]:

$$\text{Uniformity}(n, t) = \frac{1}{L} \sum_{l=1}^L r_{n,t,l} \quad (3.8)$$

- *Randomness*: This metric indicates the average randomness of the bits in the fingerprints generated from several acquisitions from a sheet. The ideal value for this metric is 1. Randomness of the fingerprint bits generated from the  $n$ -th sheet is calculated as follows [135]:

$$\text{Randomness}(n) = -\log_2 \max(p_n, 1 - p_n), \quad (3.9)$$

$$\text{where } p_n = \frac{1}{TL} \sum_{t=1}^T \sum_{l=1}^L r_{n,t,l}$$

### 3.5.2.2 Time Dimension

This dimension is concerned with fingerprint variations within multiple samples. Metrics in this dimension evaluate the overall intra-sheet persistence of fingerprints within multiple samples.

- *Reliability*: This metric shows how consistently fingerprints are reproduced by the same sheet. The ideal value for this metric is 1. Reliability of the fingerprints generated from the  $n$ -th sheet is calculated as follows [135]:

$$\text{Reliability}(n) = 1 - \frac{2}{T(T-1)L} \sum_{t_1=1}^{T-1} \sum_{t_2=t_1+1}^T \sum_{l=1}^L (r_{n,t_1,l} \oplus r_{n,t_2,l}) \quad (3.10)$$

- *Steadiness*: This metric indicates the bias of individual fingerprint bits on average for a sheet. The ideal value for this metric is 1. Steadiness of the fingerprints generated from the  $n$ -th sheet is calculated as follows [135]:

$$\begin{aligned} \text{Steadiness}(n) &= 1 + \frac{1}{L} \sum_{l=1}^L \log_2 \max(p_{n,l}, 1 - p_{n,l}) \\ \text{where } p_{n,l} &= \frac{1}{T} \sum_{t=1}^T r_{n,t,l} \end{aligned} \quad (3.11)$$

### 3.5.2.3 Device Dimension

This dimension is concerned with fingerprint variations between multiple sheets. Metrics in this dimension evaluate the overall inter-sheet distinguishability of fingerprints.

- *Uniqueness*: This metric represents how distinguishable a sheet is within a group of sheets. The ideal value for this metric is 0.5. Uniqueness of the fingerprints generated from the  $n$ -th sheet is calculated as follows [135]:

$$\text{Uniqueness}(n) = \frac{1}{T^2 L(N-1)} \cdot \sum_{t=1}^T \sum_{\substack{n'=1 \\ n' \neq n}}^N \sum_{t'=1}^T \sum_{l=1}^L (r_{n,t,l} \oplus r_{n',t',l}) \quad (3.12)$$

- *Bit-Aliasing*: This metric indicates how likely different sheets are to produce identical fingerprint bits. The ideal value for this metric is 0.5. Bit-aliasing of the  $l$ -th bit of the fingerprints generated from a dataset is calculated as follows [135]:

$$\text{Bit-Aliasing}(l) = \frac{1}{NT} \sum_{n=1}^N \sum_{t=1}^T r_{n,t,l} \quad (3.13)$$

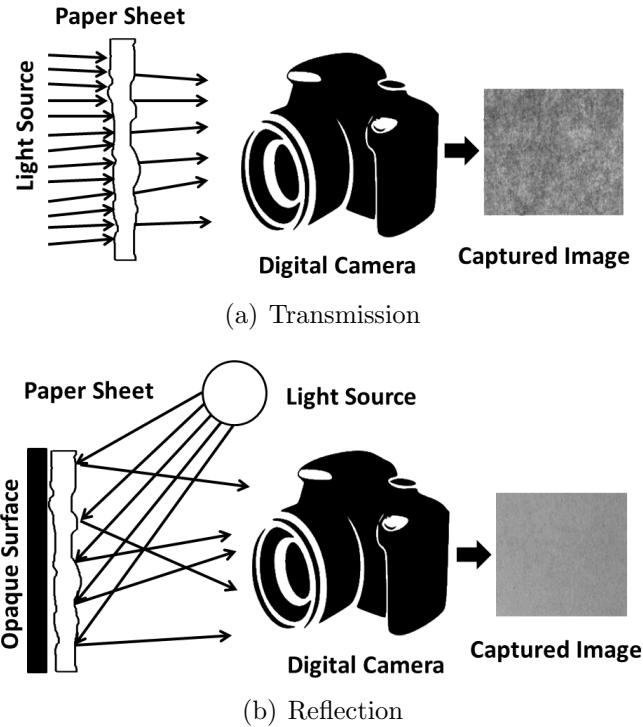


Figure 3.5: Capturing a photo, in case of (a) transmission, and (b) reflection, using the same digital camera and light source.

### 3.5.3 Reflection vs. Transmission

As discussed before, the main motivation of our work is to capture paper textural patterns and efficiently extract unique paper fingerprints from such patterns using an off-the-shelf camera. By contrast, previous works [31, 48, 178] extract paper fingerprints from the paper surface. Our hypothesis is that textural patterns revealed by the transmissive light contain richer features than the paper surface shown by the reflective light. To verify this hypothesis, we set up an experiment to investigate the difference between the two patterns.

We set up the paper photographing in two settings: one with the light source on the same side of the paper and the other with the light source on the opposite side of the paper (see Figure 3.5). In the former, we put an opaque object behind the paper, so only the paper surface is photographed based on the reflective light. We selected 10 common A4 ( $210 \times 297$  mm) paper sheets with grammage  $80 \text{ g/m}^2$ . We took 10 photos of each sheet in each of the two settings. We used a common overhead projector as our light source. We tried to reduce the effect of any ambient light by setting our data collection environment in a dark room. This data collection resulted in two datasets: a 100-sample dataset (10 sheets with 10 samples for each sheet) for

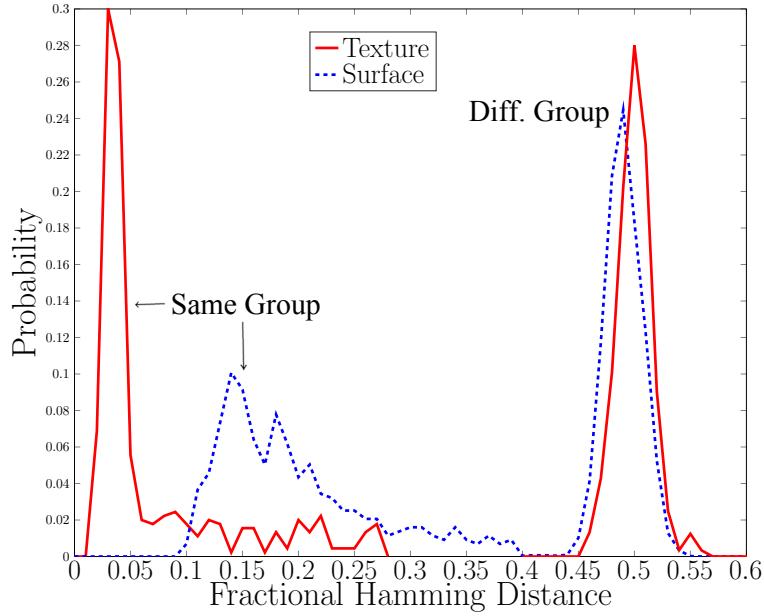


Figure 3.6: Hamming distance distributions for surface and texture.

surface measurements, and a 100-sample dataset (10 sheets with 10 sample for each sheet) for textural measurements.

After the data collection, we performed our fingerprint extraction algorithm (as discussed in Section 3.4.1) for both datasets. Figure 3.6 shows the Hamming distance distributions for the two cases. Each diagram depicts four distributions: for each case i.e., surface and texture, there is one curve, concentrated around lower values of Hamming distance, showing the distribution of Hamming distance between pairs of fingerprints of the *same* paper sheet, and a second curve, concentrated around a Hamming distance value of about 0.5, showing the distribution of Hamming distance between pairs of fingerprints of *different* paper sheets.

Ideally, for effective fingerprint recognition, we want the “same-group” and “different-group” distributions to be as separate as possible, since then we can easily decide on a threshold and consider any two fingerprints with a Hamming distance below that threshold to belong to the same paper sheet, and consider any two fingerprints with a Hamming distance above that threshold to belong to different paper sheets.

As can be seen in Figure 3.6, the two distributions, i.e., “same-group” and “different-group”, are well-separated in the case of texture, but less so in the case of surface. In fact, in the case of texture, the minimum Hamming distance for different comparisons is 0.46 and the maximum Hamming distance for similar comparisons is 0.27, which shows that there is no overlap between the two distributions. However, in the case

of surface, the minimum Hamming distance for different comparisons is 0.44 and the maximum Hamming distance for similar comparisons is 0.48, which shows that there is some overlap between the two distributions, and hence false negative or false positive decisions are inevitable in this case. Indeed, decidability for the case of texture is around 20, but for the case of surface it is around 6. Furthermore, the number of degrees of freedom provided by the texture is slightly higher than that provided by the surface. These results support our hypothesis that the textural measurements through transmissive light contain more distinctive features than surface measurements based on reflective light, and hence can be used as a more reliable source for paper fingerprinting.

We should stress that the hypothesis is tested using a specific image capturing condition in which only one snapshot is taken. One should not directly compare the results with Clarkson et al.'s 3D method [48], which is carried out in a different test condition and involves taking four scans from four different angles on the paper surface. However, we believe a method that is based on taking a single snapshot is easier and quicker than those that require multiple measurements.

### 3.5.4 Determining Gabor Scale & Orientation

As discussed, Gabor filter can be configured with different scales and orientations. To find out the appropriate combination of scales and orientation for our method, we set up an initial experiment. We collected a dataset including two sub-datasets: the first one includes 20 samples from one paper sheet; the second one includes one sample from each of 20 paper sheets. These two sub-datasets constitute our same-group and different-group data, respectively. We applied Gabor filter for 8 orientations, indexed from 1 to 8, representing angles  $0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3\pi}{8}, \frac{\pi}{2}, \frac{5\pi}{8}, \frac{3\pi}{4}$ , and  $\frac{7\pi}{8}$ . Considering  $f_{\max} = 0.25$ , we also considered multiple scales, indexed by integer values starting from scale 1. We used fixed values of  $\eta = \gamma = \sqrt{2}$  and  $\sigma = 1$ .

Ideally, we would want the different-group distribution to be centred around 0.5 or a mean very close to 0.5. Our experiments show that for scales greater than 7, the mean of the different-group distribution falls below 0.45, which indicates undesirable bias on the binomial distributions (i.e., tossing a coin is no longer random in the a Bernoulli trial [57]). Therefore, in the following we limit the scope of our investigation to scales from 1 to 7.

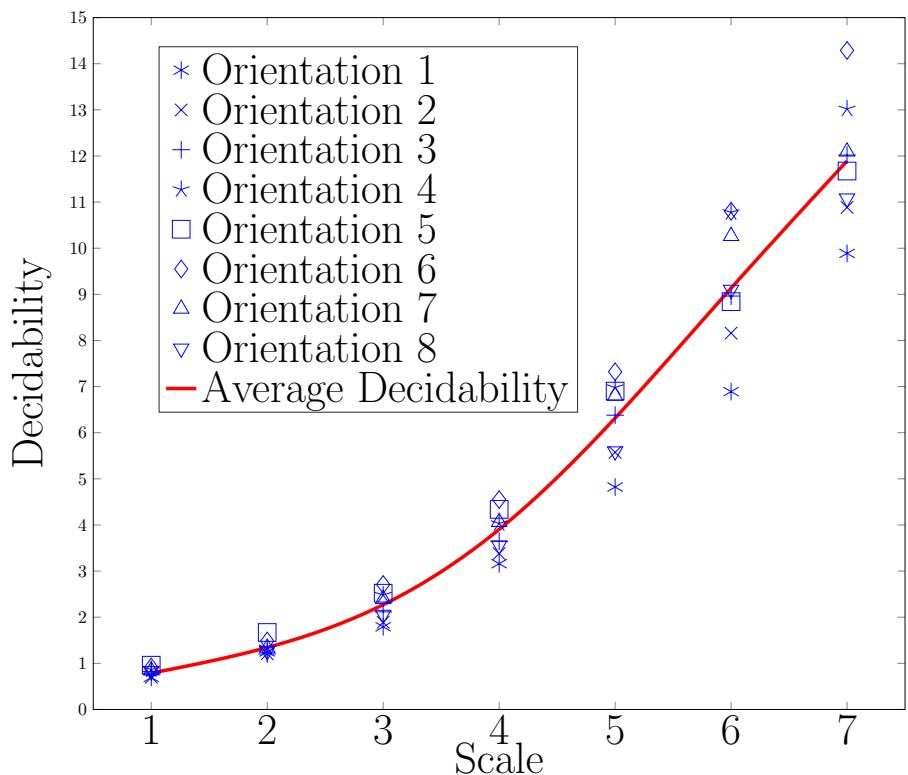
Our calculations show that as the scale increases, the decidability of the distributions increases, but at the same time the number of the degrees of freedom the different-group distribution provides decreases. This is because the scale relates to

the spatial frequency components of the Gabor filer – the smaller the scale is, the more detailed the feature extraction is. When the scale is one, the finest detail of the paper texture is extracted, which leads to high degrees of freedom in the generated fingerprint. However, at this scale, the image processing is extremely sensitive to noise, which reduces the separation between the same-group and different-group histograms of Hamming distances. Increasing the scale results in a zooming-out effect. More correlations between bits are introduced, which reduces the degrees of freedom. But on the other hand, the feature extraction is more tolerant of noise. As a result, the same-paper and different-paper characteristics become more distinctive, which leads to a higher decidability.

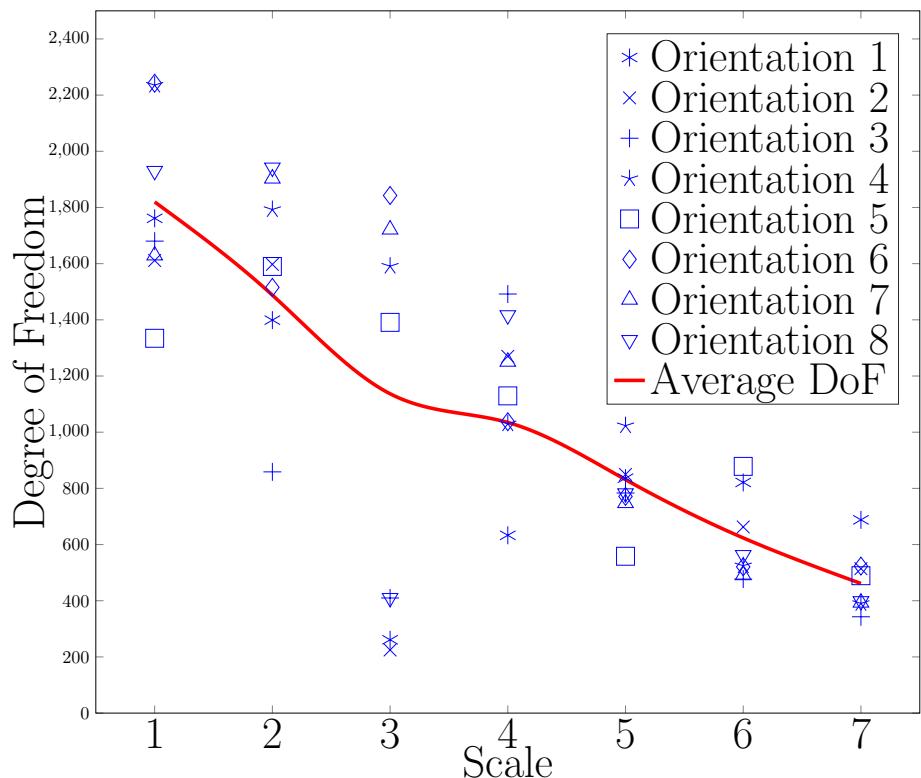
The results for decidability and degrees of freedom for orientations 1 to 8 and scales 1 to 7 are shown in Figures 3.7(a) and 3.7(b), respectively. Both figures also include a spline interpolation of average values of different orientation results within each scale to highlight the dominant trends. Therefore, there is an evident trade-off in choosing the scale and orientation. Too low a scale would not provide an acceptable decidability, while too high a scale would not provide a reasonable degree of freedom. Through experiments, we find that the combination of scale 5 and orientation 7 provides a good trade-off between decidability and degrees of freedom. As we will explain later, this combination provides nearly perfect recognition rates. In the rest of the paper, we report all our findings based on this specific configuration of Gabor filter.

### 3.5.5 The Benchmark Dataset

Our main dataset on which we report our evaluations is a set of 1000 samples collected by taking 10 photos of each of 100 different paper sheets to provide a good diversity. We use typical office paper sheets of size A4 (210mm × 297mm) with grammage of 80 g/m<sup>2</sup>. All the sheets were from the same pack with the same brand. In all of the photos, camera settings including aperture and exposure time were kept constant (exposure time set to 1/1000 seconds and F-stop to f/8). We tried to keep the paper sheets visually aligned for the different samples, and conducted separate experiments to evaluate the robustness of our algorithm against rotations (which we discuss in Section 3.6). We refer to the main dataset collected here under relatively stable conditions as the benchmark dataset.



(a) Decidability for all orientations and scales.



(b) Degrees of freedom for all orientations and scales.

Figure 3.7: Results of (a) decidability and (b) degrees of freedom in scales 1 to 7 and orientations 1 to 8.

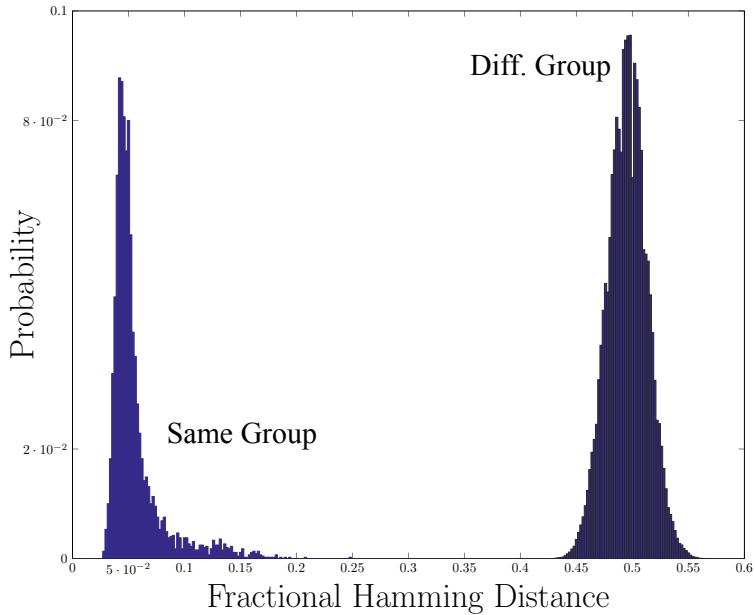


Figure 3.8: Hamming distance distributions in the benchmark dataset.

### 3.5.6 Experiment Results

In the following, we present the results of our experiments reporting the metrics introduced in Section 3.5.2. We also present the timing measurements for our method and provide a short discussion on its practicality. We provide comparison with existing work whenever the relevant metrics are reported in the literature.

**Biometric metrics.** We calculated the Hamming distance for all comparisons, consisting of same-group comparisons and different-group comparisons. There are a total of  $\binom{1000}{2} = 499,500$  comparisons, of which  $100 \cdot \binom{10}{2} = 4,500$  are same-group comparisons and  $\frac{1000 \times 990}{2} = 495,000$  are different-group comparisons. In Figure 3.8 we show the distributions for the same-group and different-group Hamming distance values. Clearly, the two distributions are well-separated, which shows the effectiveness of our paper fingerprinting method. Indeed, the maximum same-group Hamming distance is 0.24, whereas the minimum different-group Hamming distance is 0.42, which shows that there is no overlap between the two distributions.

Hence, any threshold between the above values would give us negligible FAR and FRR.

As an example, we can choose the threshold to be 0.4, but this is adjustable. Detailed error rate performance will be reported in Section 3.7.2.

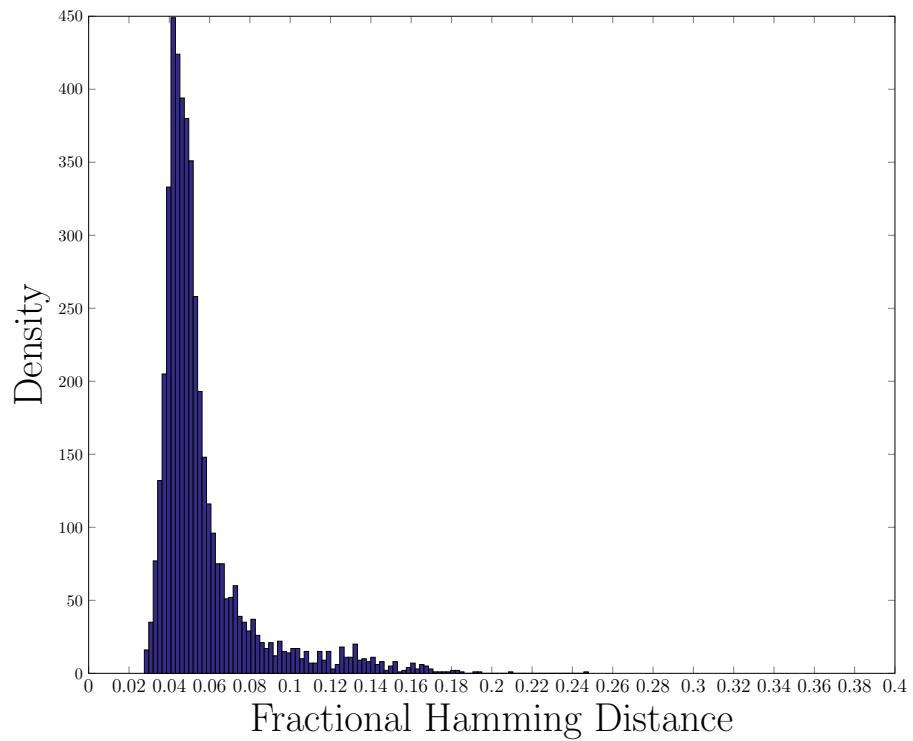
Decidability for the dataset is  $d' \approx 21$ , which compares favourably to  $d' \approx 14$  for iris recognition in the ideal condition [57]. The number of degrees of freedom

is calculated based on Equation 3.7 as  $N = 807$ , which means the entropy of the extracted fingerprints is 807 bits out of a total of 2048 bits. As compared to the 249 degrees of freedom for iris (which has the same size of 2048 bits), the fingerprint in our case is more unique and contains less redundancy. Figure 3.9 shows the histogram of same-group Hamming distance values on the left and the distribution of different-group Hamming distance values on the right. The diagram on the right also includes a binomial distribution curve with degrees of freedom  $N = 807$ , mean  $\mu = 0.495$ , and standard deviation  $\sigma = 0.018$ . Evidently, the different-group distribution closely follows the binomial distribution.

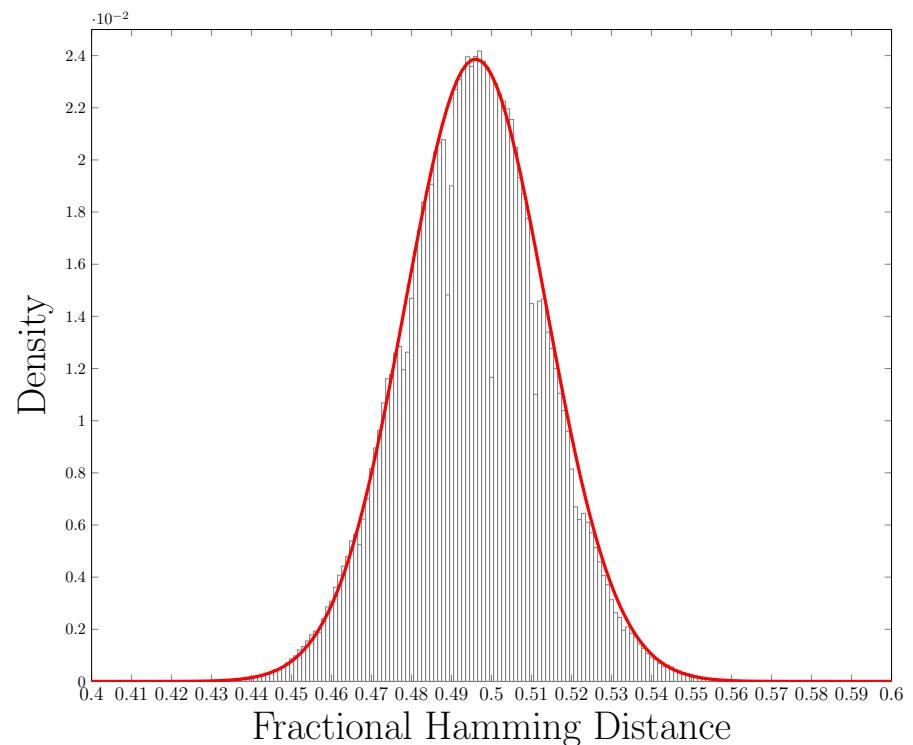
**PUF evaluations results.** The PUF metrics results on the benchmark dataset are shown in Table 3.2 under the column labelled “Benchmark Dataset”. It can be seen that in all metrics our dataset performed close to ideal values. For comparison, we also included in Table 3.2 the PUF metrics for two typical PUFs: Arbiter PUF, and Ring Oscillator PUF [135]. This shows that our method provides fingerprints with good uniformity, randomness, reliability, steadiness, uniqueness, and bit-aliasing.

**Timing Results & Usability.** Our paper fingerprinting method takes 1.30 seconds on average to prepare the photo, analyse the texture, and generate the fingerprint on a PC. This is reasonably fast. This is in contrast with the method in [48], which requires four scans in different directions and then constructing a 3D surface model. Although the authors of [48] do not report timing measurements for their fingerprinting method, 3D modelling is generally considered a computationally expensive task [27].

The whole process of paper fingerprinting in our method is automatized and only requires a user to place the sheet of paper on the flat surface of the overhead projector and click a button to take a photo by a fixed camera. We note that this is only a proof-of-concept prototype to demonstrate the feasibility of extracting the fingerprint based on the textural patterns. One may improve the prototype in a practical application by tighter integration of various equipment components. For example, at a border control, when the official swipes a page in the passenger’s passport through a slot, the slot may have the embedded light source on one side and a camera on the other side. When the page is in the slot, a unique fingerprint can be extracted. The fingerprinting area and orientation will be relatively fixed as it is determined by the dimensions of the slot. By comparing the extracted fingerprint with a reference sample (e.g., stored in the back-end system), the computer program can quickly determine if the passport page is genuine. In Section 3.7, we will explain more on how to utilizing the unique paper fingerprint in authentication protocols.



(a) Histogram of same-group HDs with  $\mu = 0.056$ ,  $\sigma = 0.024$



(b) Histogram of different-group HDs with a binomial curve with  $N = 807$ ,  $\mu = 0.495$ ,  $\sigma = 0.018$

Figure 3.9: Histograms of Hamming distances in the benchmark dataset.

Table 3.1: False recognition rates of all datasets considering a fractional HD threshold of 0.4

Rate	Ideal Value	Benchmark Dataset	Rotated	Crumpled	Scribbled	Soaked	Heated	Light Box
FAR	0%	0%	0%	0%	0%	0%	0%	0%
FRR	0%	0%	0.32%	3.2%	0%	0%	0%	0%

Table 3.2: PUF metrics for all datasets and two typical PUFs

PUF metrics	Ideal Value	Arbiter PUF (APUF) [135]	Ring Oscillator PUF [135]	Benchmark Dataset
Average Uniformity	0.5	0.556	0.505	0.466
Average Randomness	1.0	0.846	0.968	0.907
Average Reliability	1.0	0.997	0.991	0.945
Average Steadiness	1.0	0.984	0.985	0.938
Average Uniqueness	0.5	0.072	0.472	0.465
Average Bit Aliasing	0.5	0.195	0.505	0.466

Table 3.3: Impact of Robustness Experiments on PUF metrics

PUF metrics	Ideal Value	Benchmark dataset	Rotated	Crumpled	Scribbled	Soaked	Heated	Light Box
Average Uniformity	0.5	0.466	0.466	0.463	0.454	0.460	0.460	0.466
Average Randomness	1.0	0.907	0.906	0.896	0.873	0.877	0.890	0.907
Average Reliability	1.0	0.945	0.877	0.852	0.856	0.750	0.882	0.946
Average Steadiness	1.0	0.938	0.839	0.528	0.870	0.554	0.554	0.939
Average Uniqueness	0.5	0.465	0.465	0.470	0.468	0.463	0.461	0.469
Average Bit Aliasing	0.5	0.466	0.466	0.463	0.454	0.460	0.460	0.466

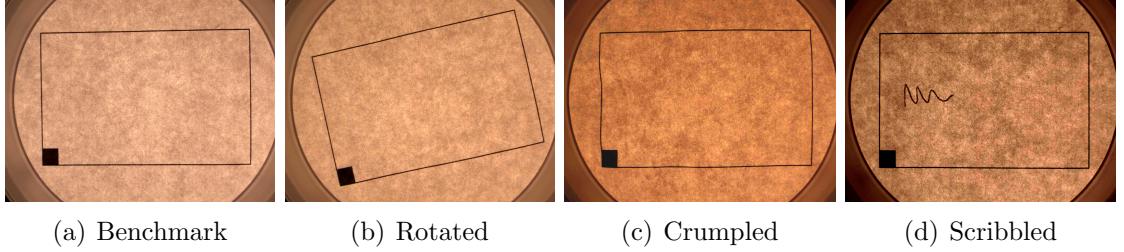


Figure 3.10: The captured photo under near-ideal and non-ideal situations.

## 3.6 Robustness Analysis

In this section we evaluate our method’s robustness in non-ideal circumstances. First, we consider the robustness of our method against misalignment, i.e., in cases where the rectangular box is not aligned to the photo frame. Then, we consider the robustness of our method against paper being roughly handled in the following cases: the paper sheet is crumpled, some scribbling is done in the rectangular box, the sheet is soaked in water and dried afterwards, and the sheet is ironed after soaking and partially burnt. Finally, we consider the effect of using an alternative light source. In the following, we give the details of each experiment and provide the biometric and PUF metrics in each of the cases.

### 3.6.1 Impact of Non-Ideal Data Collection

**Photo Rotation.** The orientation of the photo is the angle between the rectangular box and the photo frame. A rotated photo is shown in Figure 3.10(b). The maximum rotation we can have such that the box is still fully captured within the boundary of the photo frame is around  $12^\circ$ . We selected 10 paper sheets and collected 5 samples in each angle within  $\{-12^\circ, -11^\circ, \dots, 0^\circ, \dots, +11^\circ, +12^\circ\}$ . This gives us 125 samples per sheet, 1250 samples in total.

Figure 3.11 shows the Hamming distance distributions. As expected, the same-group and different-group distributions get slightly closer to each other in comparison with the benchmark dataset. However, decidability, although reduced, is still a healthy  $d' \approx 8$ . This shows that our image processing method is somewhat sensitive to the image rotation. We believe there is still room to improve the robustness against rotation, however with the current method and based on a threshold of 0.4, the FAR is still 0%, and the FRR is less than 1%. These values can be found in Table 3.1.

The PUF metrics are presented in Table 3.2. The experiment dataset still has good uniformity, randomness, and bit-aliasing, but there is a slight drop in reliability,

steadiness, and uniqueness compared to the benchmark dataset.

The experiment shows that our method is robust against non-ideal data collection in terms of rotation. In comparison, Clarkson et al. do not report robustness against rotation and in fact require “precise alignment of each surface point across all scans” [48].

### 3.6.2 Impact of Non-Ideal Paper Handling

In this section we investigate the robustness of our method against rough handling of paper sheet including crumpling, scribbling, soaking, and heating. For each of the experiments in this section, a set of 10 paper sheets are selected. For each paper sheet, 5 samples were taken before and 5 samples after the non-ideal handling of the paper sheet, adding up to a total of 100 samples per experiment. The same-group and different-group distributions under the test conditions of crumpling, scribbling, soaking and heating are shown in Figure 3.11. For readability, we opt to show fitted curves for the distributions. These curves are non-parametric fits with a threshold bandwidth of 0.02 (i.e., the distributions are merely smoothed).

**Crumpling.** In this experiment, we crumpled our paper sheets to the extent that the borders of the rectangular box were visibly distorted. We did not try to smooth out the sheet surface after crumpling. An example of a photo taken from a crumpled paper sheet can be seen in Figure 3.10(c).

The resulting Hamming distance distributions are shown in Figure 3.11. Decidability is  $d' \approx 4.6$ . Based on the threshold of 0.4, the FAR is still 0%, and the FRR is 3.2%. These values can be found in Table 3.1.

The PUF metrics are presented in Table 3.3. The experiment dataset still has good uniformity, randomness, and bit-aliasing, but there is a slight drop in reliability and uniqueness and a bigger drop in steadiness compared to the benchmark dataset.

**Scribbling.** In this experiment, we drew random patterns with a black pen over all samples such that each pattern covers around 5% of the box area. An example of such scribbling can be seen in Figure 3.10(d).

The resulting Hamming distance distributions are shown in Figure 3.11. The maximum same-group Hamming distance is 0.25 and the minimum different-group Hamming distance is 0.45. The distributions are well-separated. Decidability is  $d' \approx 9.7$ . Based on the threshold of 0.4, the FAR is still 0%, and the FRR is also 0%. These values can be found in Table 3.1.

The PUF metrics are presented in Table 3.3. The experiment dataset still has good uniformity, randomness, and bit-aliasing, but there is a slight drop in reliability, steadiness, and uniqueness compared to the benchmark dataset.

**Soaking.** In this experiment, we submerged the paper sheets in tap water for around 20 seconds. Then, we let them dry naturally and collected the after-soaking samples from the dried sheets.

The resulting Hamming distance distributions are shown in Figure 3.11. The maximum same-group Hamming distance is 0.36 and the minimum different-group Hamming distance is 0.44. The distributions are well-separated. Decidability is  $d' \approx 6.8$ . Based on the threshold of 0.4, the FAR is still 0%, and the FRR is also 0%. These values can be found in Table 3.1.

The PUF metrics are presented in Table 3.3. The experiment dataset still has good uniformity, randomness, and bit-aliasing, but there is a slight drop in reliability and uniqueness and a bigger drop in steadiness compared to the benchmark dataset.

**Heating.** In this experiment, we ironed all the papers from the soaking experiment for at least 20 seconds, to the extent that in some cases there was a clearly visible colour change (to light brown) and the paper was partly burnt.

The resulting Hamming distance distributions are shown in Figure 3.11. The maximum same-group Hamming distance is 0.30 and the minimum different-group Hamming distance is 0.44. The distributions are well-separated. Decidability is  $d' \approx 8.6$ . Based on the threshold of 0.4, the FAR is still 0%, and the FRR is also 0%. These values can be found in Table 3.1.

The PUF metrics are presented in Table 3.3. The experiment dataset still has good uniformity, randomness, and bit-aliasing, but there is a slight drop in reliability and uniqueness and a bigger drop in steadiness compared to the benchmark dataset.

**Summary.** Taking all the above results into consideration, we can see that our method shows the strongest robustness against scribbling. Both the biometric and PUF measures support this observation. The Hamming distance distributions are well-separated and all PUF metrics remain close to ideal values. Fingerprinting is also fairly robust against rotation, soaking, and heating. There is no or negligible false rejection rates and all PUF metrics possibly except for steadiness remain close to ideal values. Crumpling seems to pose the strongest challenge to robustness. Although false rejection rate is 3.2% and steadiness is not ideal, the method is still able to provide 0% false acceptance rate and healthy PUF metrics otherwise.

Focusing on biometric metrics, authentication rates remain perfect or nearly perfect under all robustness tests. This means our method provides a promising candi-

date for paper-based document authentication in practice which is able to cope with non-ideal sample collection and rough handling.

Focusing on PUF metrics, space and device dimension metrics stay close to ideal values under all tests, which indicates that the quality of fingerprint bits are still good and the sheets remain clearly distinguishable from one another. Time dimension metrics remain close to ideal values for rotation and scribbling, but steadiness and in some cases reliability drops as a result of crumpling, soaking, or heating. This is expected as crumpling, soaking, and heating physically change the paper sheets.

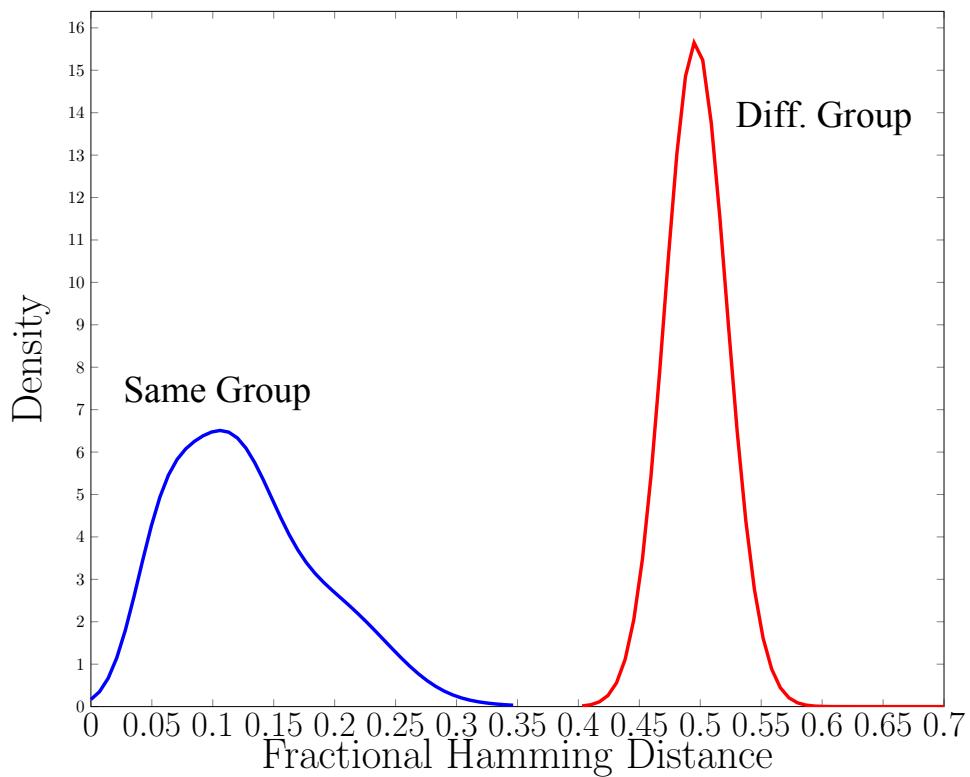
### 3.6.3 Impact of a Different Light Source

The light source should be bright enough to reveal the texture patterns in a paper sheet. In the proof-of-concept experiments, we used an overhead projector, however, the equipment is relatively bulky and expensive. Questions remain if there are cheaper ways to obtain the light source and if the results are robust against using a different light source. To investigate this, we purchased a commodity light box (tracing pad) from Amazon for £49.99 (see Figure 3.4(c)). Then, we used the same paper sheets as in the benchmark dataset—excluding 10 paper sheets that were used in other robustness tests—to collect a new set of samples using the new light source. We followed the same data collection procedure as before.

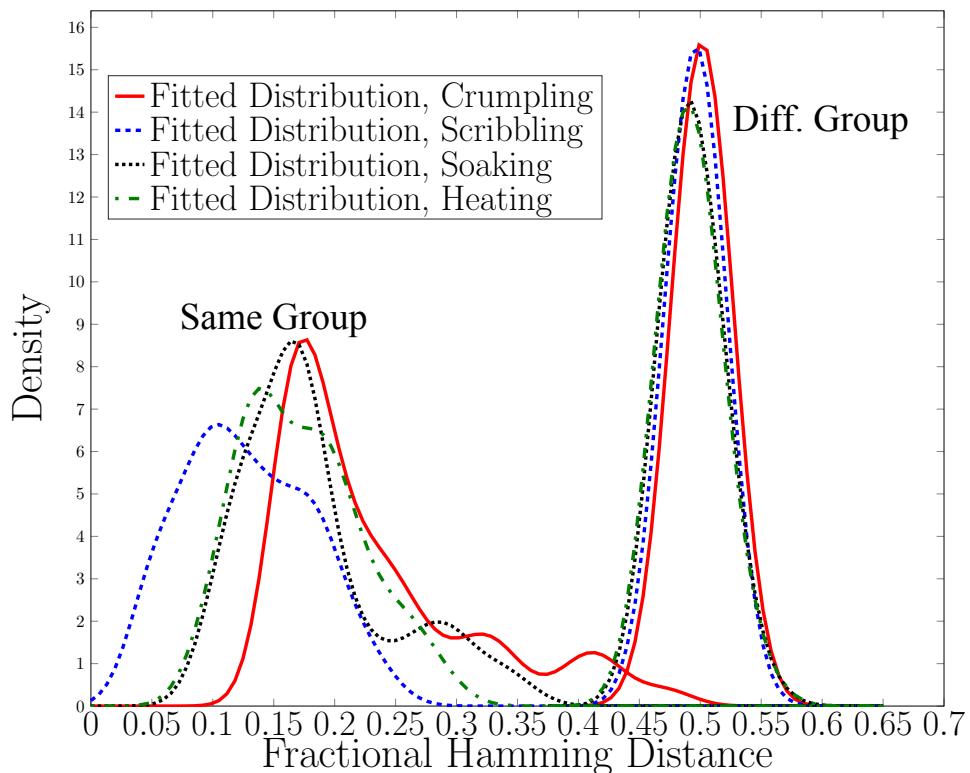
Due to the difference in the light intensity, the camera setting needs to be adjusted. In particular, we altered the exposure time to 1/500 seconds and F-stop to f/5. These values were automatically recommended by the camera, so we simply accepted them. The exposure time is the duration that the shutter takes to capture a photo and F-Stop is the radius of the lens diaphragm; both of them are inspired by the way human eyes react to a light source. These modifications in the camera setting were necessary because of the change in the intensity of the light source. The final dataset included 900 captured images, 10 samples from each paper sheet.

Figure 3.12(a) shows the Hamming distance distributions using the light box. The same-group and different-group distributions are well-separated from each other. Applying the biometric metrics, our analysis shows the decidability  $d' \approx 24$  and the number of the degrees of freedom  $DoF \approx 846$ , both slightly higher than those obtained with the overhead projector. Based on the threshold of 0.4, the FAR and FRR are still 0%. These values can be found in Table 3.1.

The PUF metrics are presented in Table 3.2. The new experiment results show that all PUF metrics are comparable to those obtained earlier in the benchmark dataset. In fact, we have a slight increase in Reliability, Steadiness and Uniqueness.

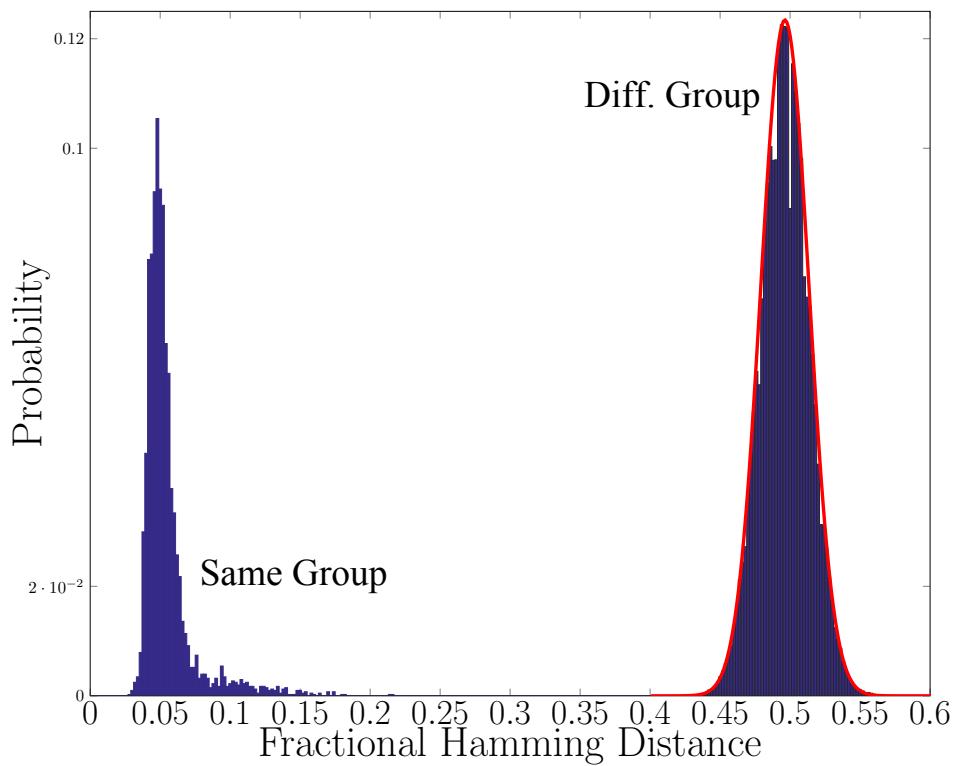


(a) Fitted distributions under rotation.

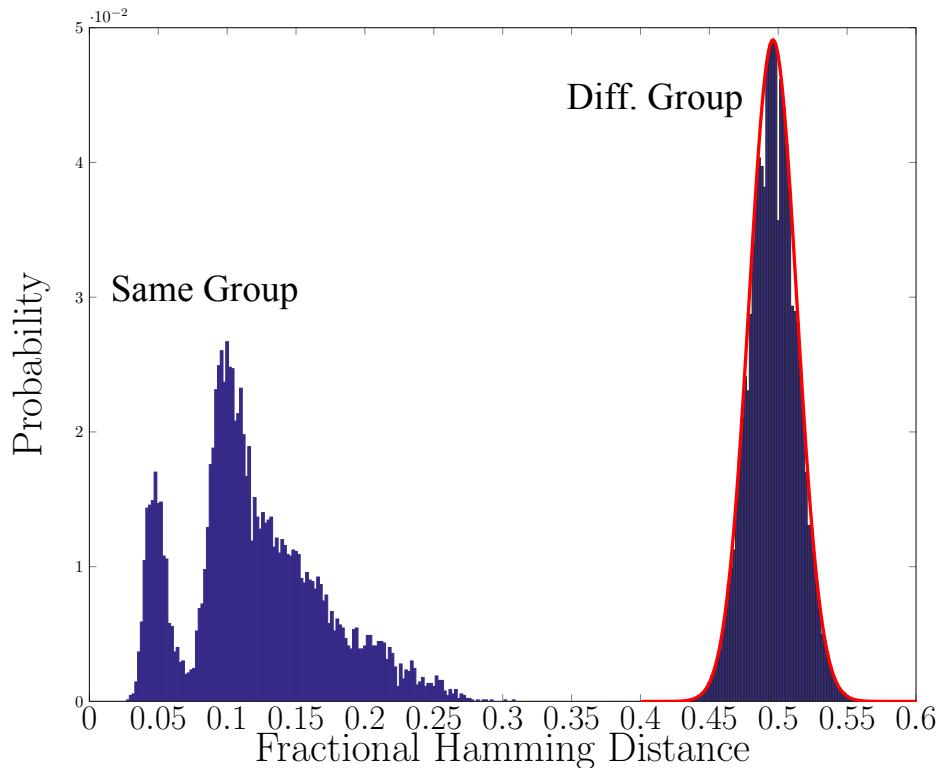


(b) Fitted distributions under non-ideal paper handling.

Figure 3.11: The Hamming distance distributions for robustness experiments.



(a) Hamming distance distributions for the light box dataset, plus the binomial curve with  $N = 846$ ,  $\mu = 0.496$ ,  $\sigma = 0.017$



(b) Hamming distance distributions for the mixed light box and projector dataset, plus the binomial curve with  $N = 836$ ,  $\mu = 0.496$ ,  $\sigma = 0.017$

Figure 3.12(b) shows the Hamming distance distribution by combining the light box and overhead projector datasets. The number of the degrees of freedom is roughly unchanged at  $DoF \approx 836$ . However, the same-group data become noisier because of mixing two different light sources. The decidability drops to 10. Despite of the mix of different light sources, the same-group and different-group histograms are still clearly separated. The maximum Hamming distance for the same-group samples is 0.31 while the minimum Hamming distance of the different-group is 0.42.

The experiment shows that our method is robust against different light sources, as long as the camera settings are set correctly.

## 3.7 Authentication Protocols

In this section, we explain authentication protocols based on the extracted paper fingerprint, and discuss their practical performance.

### 3.7.1 Trust Assumptions

Our fingerprinting technique may be applied in a range of applications, e.g., to prevent counterfeiting of paper currency, passports, certificates, contracts and official receipts. The secure use of the fingerprint is based on two assumptions. Both assumptions are generally required in biometrics and physical unclonable functions (PUF) applications.

The first assumption is physical “unclonability”. We assume it is infeasible to physically clone a paper sheet with the same paper texture. The paper texture is formed from randomly interleaved wooden particles, as a naturally occurring outcome of the paper manufacturing process. This process can not be precisely controlled. Repeating exactly the same process to produce the same paper texture is considered to be prohibitively expensive, if not impossible [158].

The second assumption is about a trustworthy measuring process. Take the human fingerprint authentication as an example. If an attacker is able to deceive the scanner by presenting a gummy finger, the security guarantee based on the “unclonability” assumption will be lost. In any biometric or PUF application, it is important to ensure that the measurement is performed on a real object and a *fresh* measurement is acquired. In practice, this is often realized through the human supervision in the process or by using specialized equipment (e.g., iris scanners with embedded liveness test). In the case of paper documents, visual inspection can be applied to check that they are made of paper and the paper fiber texture has not been tampered

with. An attacker may try to interfere with the texture measurement by printing patterns on the paper surface. Using today’s commodity printers, it seems unlikely that an attacker is able to print patterns that are precise at the pixel level under the microscopic view of a high-resolution camera (since the print head cannot be precisely controlled and each printed dot tends to be in a scattered pattern due to imperfection of the printing process; see [48]). However, when the measurement is not guaranteed to be coming from real paper texture, the acquisition process is no longer trustworthy – an attacker can at least deny the authentication by printing random patterns with strong contrast on the paper. This threat can be addressed by checking that the intended area for authentication is free from overprinting.

### 3.7.2 Comparison Based on Hamming Distance

A straightforward application of authenticating a paper fingerprint is based on comparing the Hamming distance between two fingerprints. It consists of two phases. In the first phase, a paper fingerprint, along with a mask, is extracted from the textural patterns as the template and stored in a database. In the second phase, given a provided paper sheet, the same fingerprinting algorithm is followed to output a new fingerprint and a mask. Depending on the applications, there are two types of authentication modes: verification or recognition.

Verification works on a one-to-one comparison. This assumes the reference to the stored template is known (as it is often provided by the authenticating subject). Hence, once the template is retrieved, it is a straightforward comparison between two fingerprints based on their Hamming distance as explained in Equation 3.5. This comparison determines if the presented paper sheet is the same as the one registered earlier.

By contrast, recognition works on a one-to-many comparison. In this case, the reference to the pre-stored template is unknown. Hence, the program searches throughout the database, comparing the extracted fingerprint exhaustively with each of the stored templates in order identify a match where the Hamming distance is sufficiently small. This is the same as how iris recognition works.

In terms of accuracy, the recognition mode is far more demanding than the verification mode, because the false accept rate accumulates with the size of the database. As an illustration, let  $P_1$  be the false acceptance rate for one-to-one matching in the verification mode. Assume  $P_1$  is very small. Let  $P_n$  be the false acceptance rate in

Table 3.4: False Acceptance Rate (FAR) for comparing two fingerprints

HD Threshold	False acceptance rate
0.30	$7.1 \times 10^{-31}$
0.31	$5.3 \times 10^{-28}$
0.32	$2.7 \times 10^{-25}$
0.33	$1.0 \times 10^{-22}$
0.34	$2.5 \times 10^{-20}$
0.35	$4.5 \times 10^{-18}$
0.36	$5.8 \times 10^{-16}$
0.37	$5.2 \times 10^{-14}$
0.38	$3.3 \times 10^{-12}$
0.39	$1.5 \times 10^{-10}$
0.40	$5.2 \times 10^{-9}$

the recognition mode for a database of  $n$  records.

$$\begin{aligned} P_n &= 1 - (1 - P_1)^n \\ &\approx n \cdot P_1 \end{aligned}$$

The above equation shows that the accumulative false acceptance rate in the one-to-many mode increases roughly linearly with the size of the database [57]. Hence, for the one-to-many matching to work accurately, the false acceptance rate for the one-to-one comparison must be extremely small.

For the paper fingerprints extracted in our proposal, they have sufficient entropy to support precise recognition even for an extremely large database. Based on the binomial distributions with 807 degrees of freedom, the false acceptance rates for comparing two paper fingerprints are listed in Table 3.4. If we opt to maintain  $P_n < 10^{-6}$  for the recognition mode as stated in [57], our algorithm can easily support searching a database of 3 quintillions ( $3 \times 10^{18}$ ) fingerprints at a threshold of 0.32. By comparison, for the same accuracy ( $< 10^{-6}$ ) and the same threshold (0.32), iris recognition can only support a database of only 26 iris codes. (As stated in [57], for a database of a million iris codes, the threshold needs to be adjusted downwards to below 0.27 to keep the false accept rate under  $10^{-6}$ ). Because of the much higher degrees of freedom of paper fingerprints, they can be used for the recognition application at a much larger scale than the iris biometric.

### 3.7.3 Paper Fingerprint Encryption

One limitation with the previous verification/recognition method is that the template is stored in plaintext in the database. When the plaintext template is revealed, it may cause degradation of security. This is especially the case with biometrics, since biometric data is considered private to each individual. The paper fingerprints are essentially “biometrics” of paper. One established technique in biometrics is through “biometric encryption”. Similarly, we can apply the similar technique to realize “fingerprint encryption”. We will present one concrete construction and show that because paper fingerprints have much higher entropy than even the most accurate biometric in use (iris), the corresponding encryption scheme is able to provide much higher security assurance as well.

Our construction is based on Hao et al.’s scheme [87]. This work is inspired by Juels et al. [103] and has been successfully implemented in iris recognition. It comprises two phases. In phase one, the program extracts a paper fingerprint from the paper texture as a reference  $f_a$ . It then generates a random key  $k$  (140 bits), and expands the key to a pseudo fingerprint  $f_p = \text{ErrorCC}(k)$  (a 2048-bit codeword) where ErrorCC is an error-correction encoding scheme based on Hadamard-Reed-Solomon. Our analysis shows there is a combination of block and random errors in our fingerprints; therefore, we selected a concatenated approach. The choice of 140 bits  $k$  is a balance between security (minimum 128 bit security for the secret key) and performance, as well as considering the special parametric requirements for a concatenated code scheme to work at a desired level of error correction. Subsequently, the scheme computes an encrypted fingerprint  $r = f_a \oplus f_p$ . In addition, the program computes  $h = H(k)$  where  $H$  is a secure one-way hash function. Finally, the program stores  $r$  and  $h$  in the database. Alternatively,  $r$  and  $h$  can be stored in a 2-D barcode printed on paper. The advantage of doing so is to allow authentication in the off-line mode. In this case, an additional digital signature  $s$  should be included to prove the authenticity of data in the barcode. At this stage, the original template  $f_a$  and the random key  $k$  can be safely deleted. The registration process is summarized in Algorithm 1. In Figure 3.13, we show a QR code generated from the registration phase in our prototype implementation.

The second phase is authentication. In this phase, data from the 2-D barcode is first read and the digital signature verified. A paper fingerprint  $f_s$  is extracted from the provided paper sheet. The program then computes:



Figure 3.13: Generated QR Code in the authentication protocol. This QR code contains the encrypted fingerprint,  $H(k)$  and a digital signature for both items resulting into 718 bytes (size varies depending on the type of implemented cryptographic operation).

$$\begin{aligned}
 f_s \oplus r &= f_s \oplus (f_a \oplus \text{ErrorCC}(k)) \\
 &= (f_s \oplus f_a) \oplus \text{ErrorCC}(k) \\
 &= e \oplus \text{ErrorCC}(k)
 \end{aligned}$$

In the above equation,  $e$  can be regarded as “noise” added to the codeword  $\text{ErrorCC}(k)$ . As we explained earlier, the Hamming distances between same-paper fingerprints typically range from 0 to 0.25. In the definition of the Hadamard-Reed-Solomon code, we follow the same coding parameters as in [87]. The resultant error correction code is capable of correcting up to 27% error bits in a 2048-bit codeword. Hence, by running the Hadamard-Reed-Solomon decoding scheme, the error vector  $e$  can be effectively removed, and the original  $k$  can be recovered error-free. The correctness of the decoding process can be verified by comparing the obtained  $k$  against the retrieved  $H(k)$ . This authentication process is summarized in Algorithm 2.

---

**ALGORITHM 1:** Registration

---

```

Generate Random key  $k$  ;
Generate Reference Paper Fingerprint  $f_a$ ;
Expand key  $k$  to Pseudo Fingerprint  $f_p$  ;
Calculate  $r = f_a \oplus f_p$  ;
Calculate  $h = H(k)$  ;
Calculate Digital Signature  $s = \text{Sig}(r, h)$  ;
Store  $(r, h, s)$  in a 2-D barcode ;

```

---

The key feature of the above “fingerprint encryption” scheme is that it preserves the secrecy of the fingerprint template since it forms the basis for authentication. In this way, no fingerprint template is stored in the plain form. As an example for

---

**ALGORITHM 2:** Verification

---

```
Read  $r$ ,  $h = H(k)$  and  $s = \text{Sig}(r, h)$  ;
if Signature Verification Success then
    Generate Paper Fingerprint  $f_s$  ;
    Calculate  $f' = f_s \oplus r$  ;
    Acquire  $k'$  by decoding  $f'$  ;
    Calculate  $H(k')$  ;
    if  $H(k') == H(k)$  then
        | Success ;
    else
        | Failure ;
else
    | Failure ;
```

---

comparison, without using this encryption scheme, the barcode would contain the plain fingerprint template. An attacker will be able to trivially steal the template by using a remote video camera without needing any physical access to the paper (he might use the template to create a spoofing object similar to having a gummy-finger attack in which an attacker would clone a captured finger by using a gel-based substance).

Physical spoofing attack is not applicable to PUF-based objects since the fundamental assumption of their unclonability would contract with the possibility of creating their physical replica. However, the attacker can still implement a successful spoofing attack remotely. Hence, the application of privacy preserving protocol for authentication avoids storing the texture structure in the plain text form. The goal here is to protect the paper texture from an attacker who does not have physical access to the paper sheet itself. An adversary who has access to the barcode printed on the paper can read all data including an encrypted fingerprint  $r = f_a \oplus \text{ErrorCC}(k)$ . One potential problem as highlighted in [87] is that if the fingerprint  $f_a$  contains significant correlations between bits,  $r$  may leak information about the fingerprint. The authors of [87] use the iris code as an example to illustrate that due to a high level of redundancy in iris codes, the encrypted iris code only has a lower-bound security of 44 bits. However, 44 bits security is not sufficient to satisfy high security requirements. As a result, the encrypted iris code (also called the secure sketch in the PUF literature) should not be published as public data; instead, it should be stored in a personal token as suggested in [87].

The above limitation with the iris codes does not apply in our case. Although the paper fingerprint defined in our work has the same size (2048 bits) as an iris

code, it has much higher degrees of freedom (807 as compared to 249). Following the same sphere-packing bound as defined in [87], we estimate the lower-bound security for the encrypted fingerprints as follows. Here, the lower-bound security refers to the minimum efforts required for a successful brute-force attack, under the assumption that the attacker has perfect knowledge of the correlations within the document paper sheet's fingerprint, hence the uncertainty (or entropy) about the fingerprint is 807 bits instead 2048 bits. The error correction capability for the Hadamard-Reed-Solomon code allows correcting up to 27% error bits. So in principle the attacker only needs to guess a fingerprint that is within the Hamming distance of  $807 \times 0.27 \approx 218$  bits to the correct fingerprint. Following the estimation method in [87], based on the sphere-packing bound [83], the minimum guess effort with  $z = 807$  and  $w = 218$  is calculated with the following equation:

$$\mathcal{G} \geq \frac{2^z}{\sum_{i=0}^w \binom{z}{i}} \approx \frac{2^z}{\binom{z}{w}} \approx 2^{133} \quad (3.14)$$

The above bound states that an attacker with full knowledge about fingerprint correlations and the error correction process would need at least  $2^{133}$  attempts in order to uncover the original fingerprint used in the registration and the random key  $k$ . This 133-bit security is much higher than the 44-bit security reported in [87], and is sufficient for almost all practical applications. This is possible because the paper textural patterns are far more distinctive than iris textural patterns. In iris, there exist substantial correlations along the radial structures [57]. The same phenomenon does not exist in paper texture, which explains the higher degrees of freedom in our case. This high level of security makes it possible to simply store the  $(r, h, s)$  values on a barcode instead of in a secure database. Alternatively, they may be stored in an RFID chip, and retrieved wirelessly during the verification phrase (e.g., in a e-passport application).

We evaluate the performance of this authentication scheme based on the benchmark database and are able to report perfect error rates: 0% FRR and 0% FAR. Note that this performance evaluation is slightly different from the direct comparison between two fingerprints based on their Hamming distance. The authentication is successful, only if the Hadamard-Reed-Solomon code is able to correct the errors (introduced by the XOR between two fingerprints) added to the error correction codeword, and hence recover the same random  $k$  (verified again  $H(k)$ ). The authentication protocol can only accommodate raw fingerprints, without masks (see [87]).

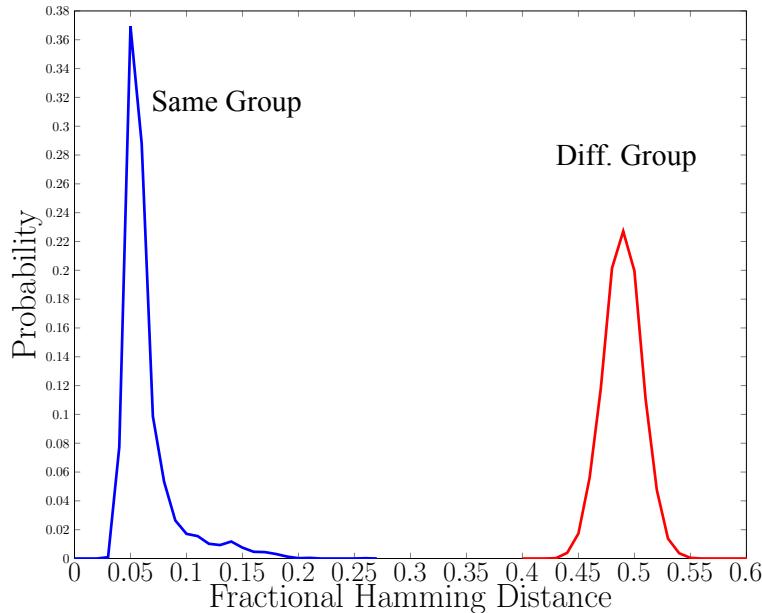


Figure 3.14: Histogram of Hamming distances between raw fingerprints without masks.

Figure 3.14 shows the histogram of Hamming distance between raw fingerprints without masks. The same-paper and different-paper distributions are well-separated. The error correcting code we implemented corrects errors up to 27%. This is sufficient to correct errors for all same-paper fingerprints, yet not sufficient for different-paper fingerprints. This explains the 0% FRR and 0% FAR that we obtain (see Figure 3.14).

### 3.8 Media Coverage

After publication of our research in this chapter, we received attention from various media outlets. The results of our research have been branded as a novel method for combating document counterfeiting and forgery. Our work has been labelled as “secure paper” in *The Economist* and “cheap and tough to crack” in *Wall Street Journal*.

Furthermore, we have received offers of collaboration from different industries world-wide. Entrepreneurs active in collectable insurance, artwork authentication and anti-counterfeiting solutions have contacted us from Hong Kong, United Kingdom, China, United States and South Africa.

The full list of our media appearance and latest news has been provided in the project’s home page: <https://toreini.github.io/projects/fingerprinting.html>.

### **3.9 Sumamry**

In this chapter, we proposed to fingerprint a paper sheet based on its texture patterns instead of features on the surface as done by previous work. We show the former contain more distinctive features than the latter with higher decidability in the histogram of Hamming distance distributions. The experiments are set up to use a commodity camera to photograph the texture patterns with a light source shining on the other side of the paper. The rich texture patterns are processed using Gabor wavelets to generate a compact 2048-bit fingerprint code. Based on the collected database, we report zero error rates, and the method is shown to work well with different light sources, and is resistant against various distortions such as crumpling, scribbling, soaking and heating. The extracted fingerprints contain 807 degrees-of-freedom, which is sufficiently high for many practical applications. As an example, some applications (like e-passport) rely on a tamper-resistant RFID chip embedded in the paper document for proving the authenticity of the document (through a challenge-response protocol based on a long-term secret stored in the chip). Our method provides an alternative solution that leverages the natural physical properties of the paper document instead of the tamper resistance of an extra embedded chip.

# Chapter 4

## Cyber Tamper Evidence: Web Page Case

In this chapter, we will introduce “*DOMtegrity*” as a mechanism to protect users from malicious browser extensions. DOMtegrity is a light-weight JavaScript framework that detects modifications to the web page’s *Document Object Model (DOM)* in a client’s browser. To our knowledge, DOMtegrity is the first proposal that provides in-browser client security against extension attacks without modifying browser’s architecture or strengthening extension’s vetting process. Instead, DOMtegrity offers protection of web page’s DOM integrity by embedding into web page’s source code via an in-line `<script>` tag. It offers a more portable solution to the long-existing problem.

DOMtegrity records the modifications occurred when the page is being rendered in the browser. Then, it securely sends the recorded changes to the server. DOMtegrity server analyses the received information and decides whether the page is tampered with or not. We successfully implement our protocol to detect a range of real-world attacks on online banking systems.

Furthermore, we will examine 14,000 real-world extensions to observe the compatibility of DOMtegrity with them. We will analyse the range of modifications each real-world extension performs. Then, we will analyse an extensive range of attack scenarios by a malicious extension. We will discuss how DOMtegrity can guarantee protection against these attacks.

DOMtegrity can be used in sensitive web pages such as online banking web sites. It can provide servers with more comprehensive insight in order to actively protect their clients against malicious extensions.

## 4.1 Introduction

Browsers extensions have become the dominant method to extend browser functionality. All major browsers (Chrome, Firefox, Safari, Opera and Internet Explorer) support extensions, and host dedicated repositories (“stores”) from which extensions can be downloaded and installed directly from the Internet. Mozilla reports average rates of more than 1 million Firefox extensions downloaded daily and about 100 new extensions created every day throughout 2016 [143].

Extensions are normally distributed and executed in controlled environments. All extensions uploaded to a repository are subject to a *vetting process*, which is a mixture of automated program analysis and manual code review and aims to identify malicious extensions and prevent their spread. Furthermore, extensions are run in a restricted (so-called “sandboxed”) environment and only have access to a predefined set of browser APIs.

However, the vetting process is not bullet-proof. A study conducted by Google researchers found nearly 10% of extensions examined to be malicious [99]. By using obfuscation, some malicious extensions can slip through the vetting process. Furthermore, the extension update mechanism provides an additional exploit path for the attacker. In 2014, two popular and previously vetted Chrome extensions, “Add to Feedly” and “Tweet This Page”, were sold to spammers who updated the extensions to inject advertisements and affiliate links into websites opened in the browser.

**The Problem.** The key problem with extensions is that, once installed, they possess over-privileged capabilities that may be abused by attackers. For example, an extension is free to modify the *Document Object Model (DOM)* of a web page. This allows a malicious extension to manipulate the display of a web page and deceive users into believing something false. The change of the web page content may be subtle, but when it is combined with social engineering techniques, it can cause significant harm to user security [70]. In Section 4.3, we will demonstrate two attacks on real-world banking websites (HSBC and Barclays) to show how a malicious extension may stealthily steal money from the user’s bank account by making small modifications to the DOM structure of an online banking web page.

Existing solutions to prevent malicious extensions generally involve changing the browser’s internal design [192, 62, 219], strengthening the vetting process of repositories [99, 105, 104, 78, 12], asking users to install yet another (trusted) extension that detects malicious behaviour of other extensions [138, 124] or requiring an external hardware device (e.g., Cronto) that performs out-of-band transaction verification.

**Our solution.** In this section, we propose a cryptographic protocol that we call *DOMtegrity* to ensure the integrity of the DOM structure of a web page delivered from a web server to the rendering of the page at the client browser in the presence of malicious extensions. Compared to previous solutions, ours does not require changing the browser’s existing internal design; it does not need any external hardware device; it is orthogonal to the strengthening of the vetting process; it can be easily implemented by embedding in-line JavaScript code in the web page rather than requiring the user to install another (trusted) extension. The novelty of our solution lies in exploiting subtle but important differences between extensions and in-line scripts in terms of their rights to access Websockets established between the server and the client. This is combined with leveraging the latest Web Crypto API that is recently added in all major browsers.

## 4.2 Related Work

This section reviews related work on countering the threats imposed by malicious browser extensions. Existing countermeasures can be categorized into four types: modifying browsers, strengthening the vetting process, requiring another trusted extension and using external hardware. Each of these types, including selected prominent examples, is reviewed in detail below.

**Modifying Browsers.** Proposals in this category require their system to be integrated natively within the browser. Ter Louw et al. design systems for protecting code integrity and user data [192]. The latter is a mechanism that augments the browser to support policy-based runtime monitoring of extension behaviour. The goal is to protect sensitive user data from being accessed or modified by the extension. Dhawan et al. proposed “Sabre”, an in-browser information flow monitor to detect malicious activities of JavaScript based extensions during runtime [62]. Sabre associates an appropriate label to all in-memory JavaScript objects based on whether they carry sensitive information. Then, it monitors the objects carrying sensitive information for any insecure access. Wang et al. proposed an extension access control framework [219], which dynamically analyses the behaviour of extensions at runtime and controls policies to restrict their access to resources. All the proposals in this category require modification of browser code base. Unfortunately none of these proposals have been adopted by mainstream browsers so far. In fact, some of these proposals are based on the XPCOM model for creating extensions in Firefox which is due to be deprecated in favour of WebExtensions.

**Strengthening the Vetting Process.** Proposals in this category involve various techniques to improve detection rates of malicious extensions during the vetting process. Jagpal et al. shared their three years of experience in fighting with malicious browser extensions in Chrome Web Store [99]. They developed a detection system called WebEval to vet the extensions in the market. WebEval combines both static and dynamic analysis of the source code, as well as taking into consideration of the reputation of the extension’s developer, and involving human experts in manual reviews whenever necessary. Their method was able to identify real-world malicious extensions with a success rate of 96.5%.

Besides methods adopted by the industry, academic researchers also propose various techniques to strengthen the vetting process. Kashyap et al. proposed a framework to automate the vetting process in official extension repositories [105]. They proposed a notion of add-on security signature which provides detailed information on its data flow and API usages. Kapravelos et al. presented Hulk as a dynamic analysis system to detect malicious extensions [104]. They monitored the execution and network activities of extensions to detect their malicious intentions. They had an extensive collection of real-world extensions from Chrome Web Store, and one of their findings was discovering a malicious extension that affected 5.5 million users. Guha et al. proposed an IBEX framework for authoring, analysing, verifying, and deploying secure browser extensions [78]. They suggested a high level programming language to develop extensions. They also proposed Datalog to specify fine-grained access control to restrict the extension’s access to security-specific web content. Bandhakavi et al. presented the VEX framework for highlighting potential security vulnerabilities in browser extension [12]. They applied static information-flow analysis to catch malicious JavaScript code in the extension implementation.

**Requiring another Trusted Extension.** Proposals in this category require users to trust one particular extension and install it consciously. Marouf et al. proposed a run-time framework called REM that monitors the access made by extensions and provides customized permission [138]. They developed an extension for monitoring other extension based on REM. They monitored API calls from an extension to the browser and enforced their policies on the extension. They notified users about the latest activities of other extensions and allowed them to block future such activities. Liu et al. demonstrated the same threat in Chrome [124]. They also implemented an extension to enforce more fine-grained privileges to extensions in Chrome. They proposed HTML elements to use another attribute called “sensitivity” to differentiate DOM elements and enforce the policy that they call micro-privilege management.

**Using External Hardware.** Cronto<sup>1</sup> is a commercial hardware-based solution to address MITB attacks specifically for online banking. It was initially developed by a spin-off company from the University of Cambridge in 2005 and was later acquired by VASCO Data Security International for £17m in 2013. The product has been widely deployed by major banks in Chile, Switzerland and Germany to secure online banking. The Cronto solution works by using a special client device, which shares a secret key with the sever. When the user performs transactions during online banking, the server sends a 2-D barcode to display on the client’s web page, which encodes the encrypted transaction details such as the amount, timestamp and account number. The 2-D barcode is then read and verified by the Cronto device that has the decryption key. Upon successful verification, Cronto generates a one-time password (OTP), which the user can enter in the browser to authenticate the transaction. Here, the Cronto device can be either custom-built hardware with an embedded camera or a smart phone.

DOMtegrity is similar to Cronto in preventing malicious modifications on the client side against MITB attacks. However, ours is a JavaScript-based *software* solution and does not require an external hardware token. We note that although the main design aim of Cronto is to ensure the integrity of transactions, it has a secondary function as a second-factor for authentication since the device has a shared secret key with the server. DOMtegrity does not have this function, but it can be used in combination with any existing two-factor authentication scheme, e.g., the Chip Authentication Program (CAP) currently used by HSBC and Barclays.

**Other Related Work.** Reis et al. proposed the idea of ensuring web content integrity by JavaScript [167]. Their method was inspired by the Linux integrity check and AEGIS [6]. In their approach, they developed a client-side JavaScript framework named TripWire. It detected unexpected modifications (in flight modification in their terminology) done by ISPs and other intermediate nodes over HTTP communication. Once the page rendering is complete, the code requested the page’s source code from the server through AJAX requests, then the internal source code is compared with the server’s one at the client side. Tripwire did not consider browser extensions in their attack model because it considers them as “trusted”. They discussed that their method was comparable to HTTPS with better performance.

---

<sup>1</sup><https://www.vasco.com/products/two-factor-authenticators/crontosign.html>

## 4.3 Malicious Extension Attacks on Online Banking

Attacks caused by malicious extensions are often known as man-in-the-browser (MITB) attacks. To demonstrate the importance of understanding the threats imposed by malicious extensions in modern browsers, we show two proof-of-concept attacks on real-world banking websites, HSBC and Barclays, by exploiting the capability of browser extensions to modify the DOM of a web page. The extensions are developed for both Firefox and Chrome based on the standard WebExtensions framework. In the proof-of-concept demonstration of the attacks, the money was transferred between the authors' accounts.

Newcastle university regulations require any research with direct relation to sensitive data to go through the ethical review process [146]. The appointed ethical committee investigated the project in detail and approved the implementation of a proposed experiment accordingly. All the experiments that we performed in this research were conducted using our own bank accounts, and were approved by our university's ethics committee.

### 4.3.1 WebExtensions Capabilities

Before describing the attacks, we should first explain WebExtensions<sup>2</sup>. The WebExtensions framework is a cross-browser architecture for developing browser extensions using HTML, CSS and JavaScript. It is now supported in all major browsers except Safari.

An extension developed based on WebExtensions consists of three components: the *background page*, the *UI pages*, and the *content scripts*. The background page is in charge of long-term operations that last beyond the lifetime of a particular browser window and is provided with access to browser APIs. The UI pages put together the extension user interface. Content scripts are JavaScript programs that are run in the context of a web page and are allowed to interact with the page.

Although the background and UI pages do not have access to the DOM of the page, content scripts can modify the DOM. Through content scripts, an extension can hide elements of the DOM and insert another element in the same location to effectively replace the original element. For example, a text box can be placed by a malicious extension in place of a password text box to capture a user's password.

---

<sup>2</sup><https://developer.chrome.com/extensions/overview>

In the following demonstration, we assume that a malicious extension is already installed on a client’s browser. This can be done through disguising malicious extensions as legitimate browser extensions, using Trojans to install such extensions, missing plug-in attacks, or purchasing popular extensions and then adding malicious code during updates [174, 70]. In both attacks, the web pages that are presented to the victim are from the genuine banking websites via HTTPS.

#### 4.3.2 HSBC Attack

The first attack shows how a malicious extension can easily bypass the two-factor authentication that is adopted by major banks, including HSBC. In this attack, the extension intercepts the victim’s authentication credentials (i.e., login details), sends them to a remote attacker and redirects the user to a false maintenance page. Depending on the security policy of the banking web site, this authentication could involve a regular password and an additional one-time password (OTP) as a second factor which is either sent to the user’s mobile phone as an SMS or locally generated using a dedicated device (i.e., a Chip Authentication Program (CAP) device) provided by the bank.

We developed a proof-of-concept attack that targets the HSBC online banking web pages. To authenticate their clients, HSBC uses a password-based user authentication augmented with an OTP generated by a dedicated device, the HSBC Physical Secure Key. Our attack works as follows:

1. When the victim requests the login page, the browser extension content script replaces the username and password text boxes with its own and records the victim’s username and password by communicating with the extension background page.
2. When the victim is prompted for an OTP, the browser extension records what the victim enters in a similar manner.
3. The victim is then redirected to a genuine customer service page. However, the content of the page is changed on the fly by the extension content script to include a message indicating that the website is temporarily unavailable for maintenance or due to technical difficulties as shown in Figure 4.1.
4. The stolen login credentials are sent to the attacker who can then log into the victim’s online banking account.

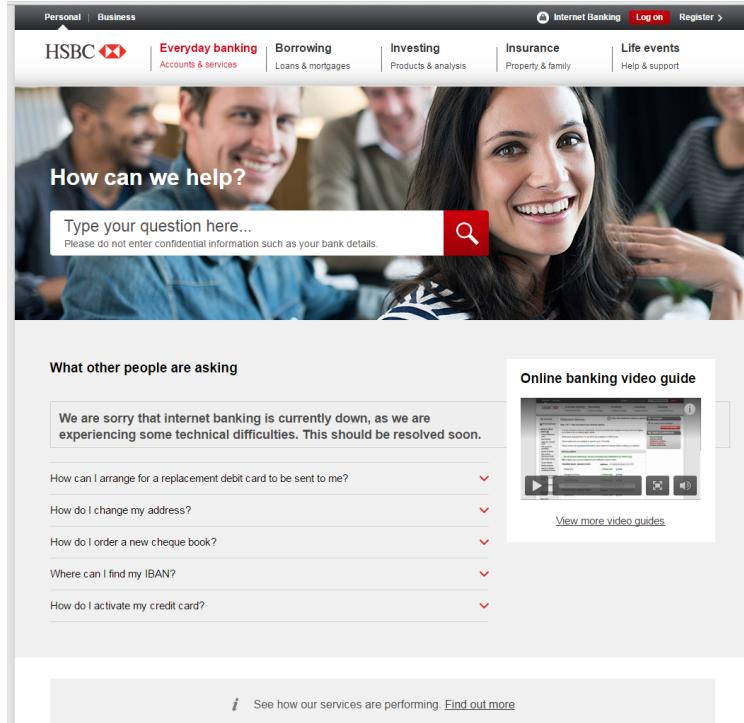


Figure 4.1: The HSBC customer service page modified by the malicious extension to contain a message indicating website technical difficulties.

We have implemented the attack by developing extensions for both Firefox and Chrome based on WebExtensions. Our extensions were able to perform the attack successfully without being detected by the bank server. Consequently, we were able to impersonate the victim and log into his or her bank account on a separate machine.

### 4.3.3 Barclays Attack

The second attack shows how a malicious extension can defeat transaction-specific user authorization, which is added by many banks such as Barclays as an extra layer of security on top of two-factor authentication. Here, when an already authenticated user requests a transaction, she is required to provide a transaction-specific authorization code which is either sent to the user out of band or generated by a dedicated device upon unique transaction-specific input. This *transaction authentication* is designed to prevent modification of transaction data (e.g., recipient and amount) by man-in-the-browser attackers.

Barclays uses the strongest form of transaction authentication (the so-called *full transaction authentication* [2]) in which the unique transaction authorization code (i.e., the transaction-specific OTP) is cryptographically bound to the transaction

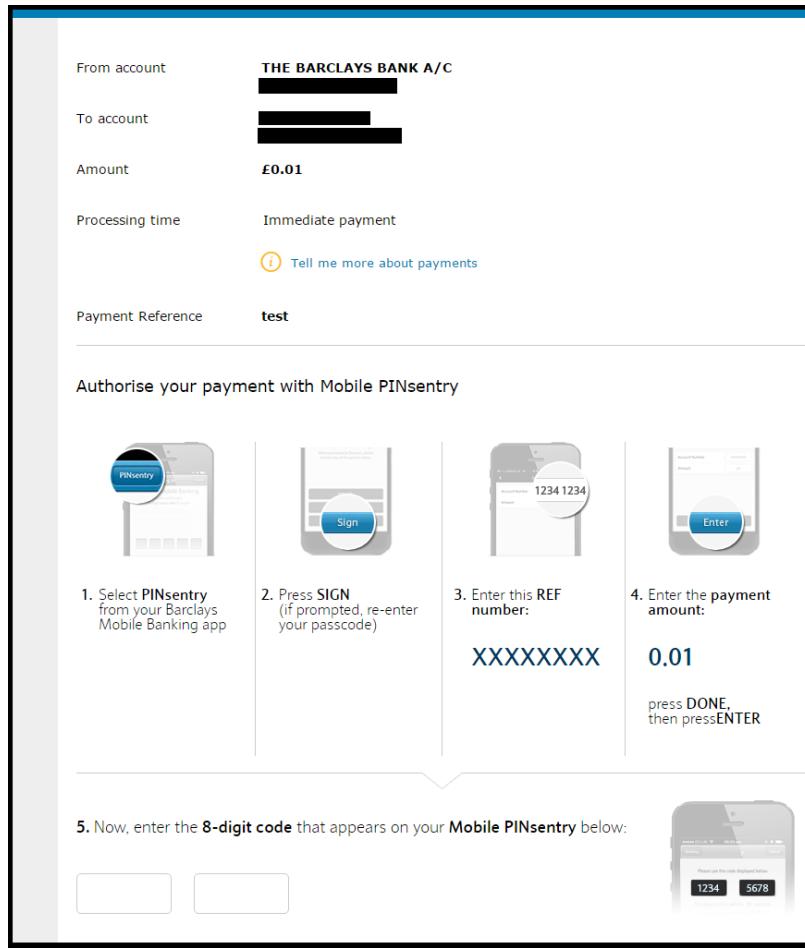


Figure 4.2: The Barclays instructions page modified by the malicious extension to include the attacker’s account number (redacted as XXXXXXXX) as the REF number.

data. The authorization code is calculated by a dedicated device provided by Barclays called PINsentry. Alternatively, the user can use the functionally equivalent Mobile PINsentry application on her smartphone. PINsentry is a battery-powered device consisting of a numeric keypad, a small LCD screen, a card reader and a processor. When a transaction is requested through Internet banking, the user is required to manually enter the transaction details, including the payee account number and the amount, on PINsentry (or Mobile PINsentry) and then enter the PINsentry produced authorization code on the internet banking web page. However, in the following we show how a malicious extension can defeat this security measure by combining social engineering and DOM modifications. The attack works as follows:

1. When the victim requests a funds transfer, she is presented a form to provide the details of the funds transfer, including the payee account number and the

amount. The malicious extension content script replaces the text box where the victim is supposed to enter the account number of the intended payee with its own text box and records the entered account number by communicating with the extension background page.

2. Then the user is presented with a dialogue confirming the transaction details and instructing her how to get a transaction authorization code from PINsentry. The instructions include asking the user to “Enter the payee’s account number as your REF:” followed by the payee’s account number. The malicious extension content script replaces this instruction with “Enter this REF number:” followed by the attacker’s account number, as shown in Step 3 of the instructions in Figure 4.2 with real bank details suitably redacted.
3. A non-expert user, trusting the HTTPS page to be secure and failing to notice the above subtle change, then enters the attacker’s bank details in PINsentry and provides a code authorizing the funds transfer to the attacker’s account.
4. The browser extension changes the final confirmation page before it is displayed to the user so that it shows the account details of the original intended payee rather than that of the attacker.

In this attack, the attacker’s bank account may eventually be discovered by checking the victim’s bank transaction records. However, this is not an issue for the attacker since he only needs to prevent the discovery of the fraud for some short timescale in which the funds can be withdrawn from the account [63].

The key issue that we were able to exploit is that PINsentry prompts the user for two pieces of transaction information: “REF” and “Amount”. The only information about what “REF” means is present on the website, which can be modified by the extension. We have responsibly disclosed our attack to Barclays and since then Mobile PINsentry has been updated and the prompt on the app has been fixed to explicitly ask the user for the payee’s account number instead of a REF number.

Recently, the second Payment Services Directive (PSD2) enforces a new concept as “Strong Customer Authentication (SCA)” to augment the security of payments within the Europe zone. At the time of writing this dissertation, the PSD2 was due to be implemented from January 13th 2018; however, the SCA compliance will not be in effect until 14 September 2019. This is mainly because the standard authentication protocols have not been agreed upon yet [14]. As a result, the authentication procedure for both online banking systems discussed above might change in the future,

which might pacify our current attacks. Nevertheless, the online banking clients will still be exposed to various attack vectors because browser extensions are considered as a trusted component of a running browser and thus, they can freely manipulate sensitive information such as DOM elements, forms inputs, session cookies and HTTP(s) requests.

## 4.4 Our Proposed Solution: DOMtegrity

In this section, we propose a solution, called DOMtegrity, to address MITB attacks such as those demonstrated in the previous section. Our solution is designed based on the WebExtensions framework, which is now the standard extension development architecture recommended by W3C and adopted by Google Chrome, Mozilla Firefox, Microsoft Edge and Opera.

### 4.4.1 WebExtensions Security Model

The WebExtensions security model as implemented in modern browsers is based on the model proposed by Reis et al. [166] who discussed the real-world security issues experienced by Google Chrome and advocated a systematic method to prevent these attacks. Here we discuss parts of this model that are necessary for the description of our protocol.

**Browser Zones.** In modern browsers, the execution environment is divided into two zones: an unprivileged *Internet zone* in which web pages are executed, and a privileged *Chrome zone* in which extensions are executed. A schematic representation of these zones is shown in Figure 4.3. Scripts in the Internet zone (i.e., the so-called *in-line* scripts within the web page) cannot have access to the data in the Chrome zone (i.e., the extension scripts), and vice versa. Therefore, although the web page scripts and the extension content scripts can interact with DOM separately, they cannot interact with each other. This concept is called the *isolated worlds* principle. The main reason for the isolation is to prevent malicious in-line scripts from exploiting the vulnerabilities that may exist in extension content scripts [4]. However, as we will explain, the isolation is also useful in defending against malicious extensions when the in-line scripts are from a legitimate source.

**Permissions.** Every extension must provide a “manifest” in the JSON format which defines the resources and the corresponding *permissions* for each component of the extension. Based on this manifest, users are prompted to grant the required

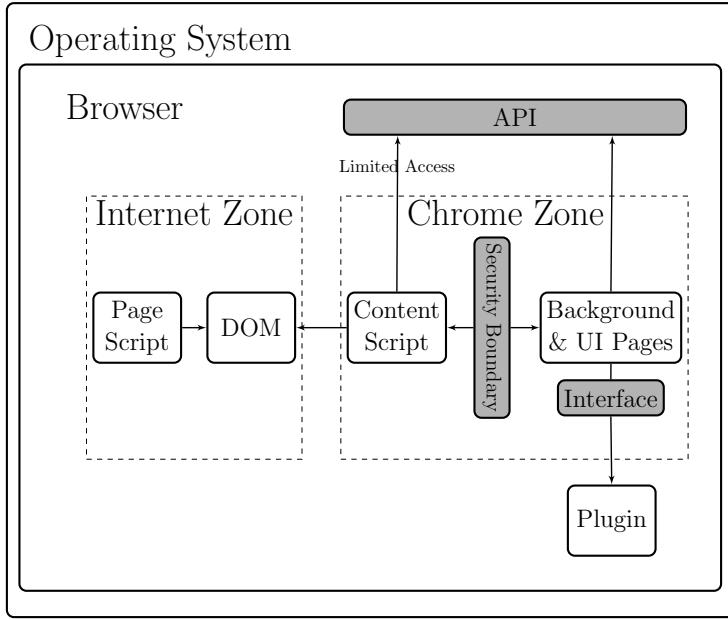


Figure 4.3: The Internet and Chrome zones of a modern browser and how web pages, extensions, and plug-ins interact [4]

permissions at the time the extension is installed, and once installed, the extension's access to browser APIs is limited to these permissions.

#### 4.4.2 Design Overview

DOMtegrity is designed to enable the server to detect any unexpected modification of the DOM by extensions when the web page is rendered in the browser. The underlying idea is that DOMtegrity securely records all the modifications made to the web page DOM until the final rendering of the page and then securely communicates the recorded modifications to the server. The server is then in a position to decide whether or not the client's browser has parsed the page as the server expected.

DOMtegrity is implemented as a JavaScript program, called `pid.js`, which is then embedded as an in-line script (within a `<script>` tag) in the web page that the server wishes to protect. This in-line inclusion is necessary since extensions are not able to restrict the execution of in-line web page scripts, whereas they can block loading external script files. For the in-line Javascript to work, we assume that JavaScript execution is not disabled in the browser.

Since DOMtegrity is to record all modifications to the DOM, it is essential that `pid.js` is placed at the start of the page source code and before all other HTML tags. Since parsing the web page in browser proceeds in the order that tags are placed in

Table 4.1: Capabilities of extension and in-line script (W3C [214]).

Capability	Extension	<code>pid.js</code>
Access the DOM	✓	✓
Establish Websockets	✓	✓
Block Websocket Establishment	✗	✗
Block Websocket communications	✗	✗
Access an expando created by <code>pid.js</code>	✗	✓
Access/close Websockets established by <code>pid.js</code>	✗	✓
Access/close Websockets established by the extension	✓	✗

the page source code, placing `pid.js` at the start of the page ensures that recording changes in the DOM starts immediately as the browser starts parsing the page.

The isolated worlds principle guarantees that DOMtegrity’s recording of modifications in DOM cannot be tampered with by any extension. When executed, `pid.js` creates an on-the-fly DOM property (also called a DOM expando) named `document.pid` which implements the DOMtegrity functions within a domain isolated from any extension.

DOMtegrity uses the recently introduced *Websocket*<sup>3</sup> technology which provides a full-duplex communication channel over TCP (or SSL/TLS for an encrypted channel) and is now supported by all major browsers. In this section, we only consider Websocket established over the secure SSL/TLS channels. The important property here is that although both in-line scripts and extension content scripts can establish Websockets, neither has access to Websockets established by the other.

The extension’s inability to access Websocket communication established by DOMtegrity provides assurance on the integrity of the communication between `pid.js` and the server. The in-line script `pid.js` establishes a Websocket with the server and this Websocket is used as a secure channel to convey a secret key which is later used to authenticate the DOM modifications that `document.pid` records. We should emphasize that although an extension has extensive access to HTTP(S) communications, it can only access the Websockets that are established by the same extension.

Table 4.1 summarizes the relevant capabilities of extensions compared with in-line scripts such as `pid.js` based on the latest W3C specification (23 July 2017) [214]. Both can access the DOM and establish Websockets, but neither can block Websocket communications. The extension cannot access the expando created by `pid.js`. Neither `pid.js` nor the extension can access or close Websockets established by the other. (From Chrome 58 introduced in April 2017, a new capability has been added in the extension API to block the initial establishment of a WebSocket channel by an in-line

---

<sup>3</sup><https://www.w3.org/TR/Websockets>

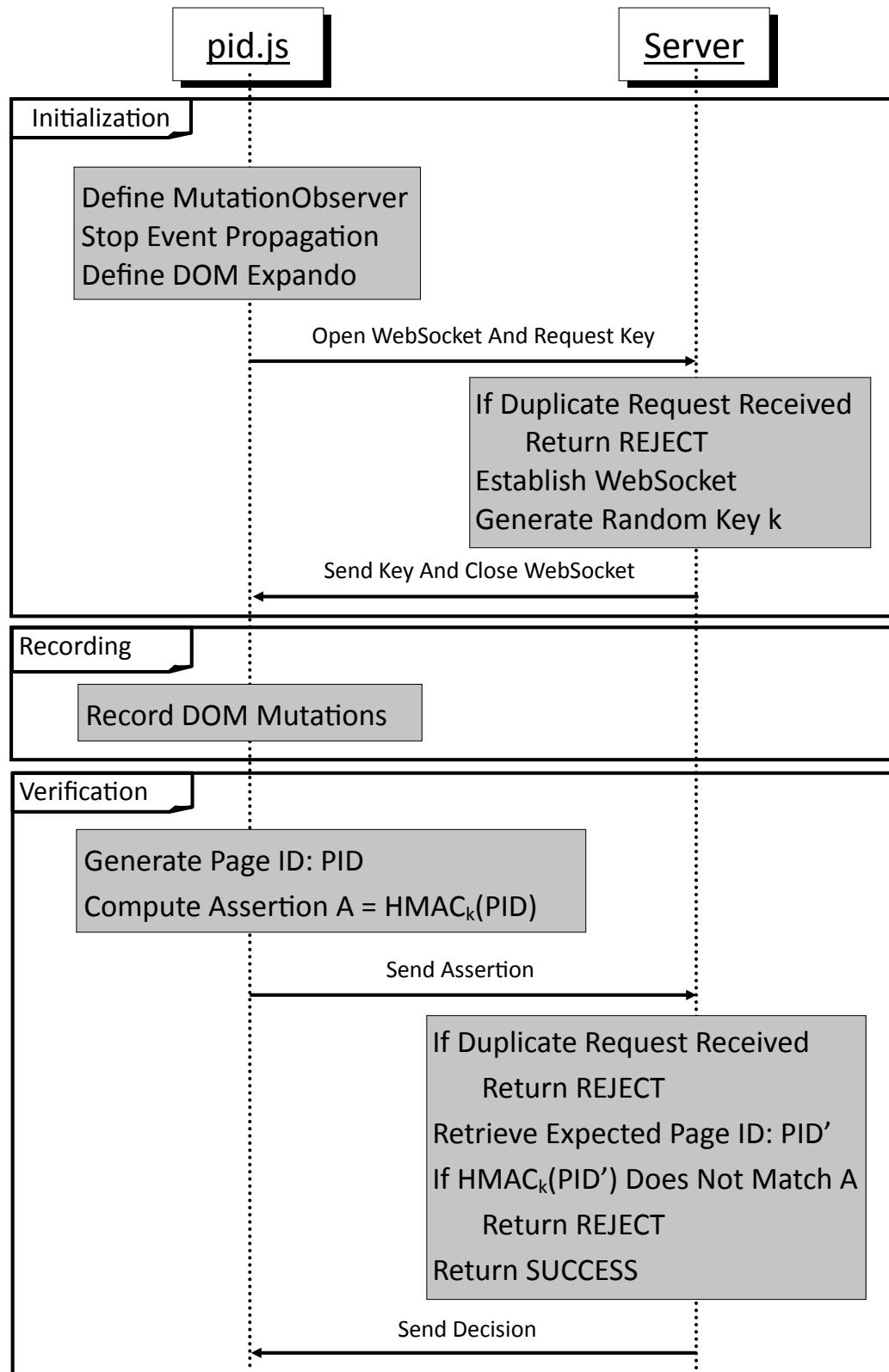


Figure 4.4: sequence diagram for DOMtegrity

script, though the W3C specification [214] has remained unchanged. We will discuss this new capability in Section 4.5, and show that it does not affect the working of DOMtegrity.)

DOMtegrity is only applicable to the *original* browser executables. We do not consider the possibility of malicious modifications to the browser internal functioning through malware in our threat model. This scenario is considered as out-of-scope since the browser extensions can not perform such attack without the need to locally install malicious software in the victim’s computer. Thus, any solution for such attack should operate in the Operating System level as well as inside the browser.

#### 4.4.3 Detailed Description

DOMtegrity runs in three stages: initialization, recording and verification. The initialization stage sets up the protocol, the recording stage is in charge of storing all DOM modifications, and eventually in the verification stage evidence of DOM integrity is generated on the client side and is sent to the server for verification. These stages are described in detail in the following. A sequence diagram of the protocol is shown in Figure 4.4. We assume the web page is served over HTTPS. The client is identified by the TLS session ID.

##### Stage 1: Initialization

This stage begins as the browser starts parsing the web page. In this stage, the required setup for DOMtegrity is carried out as follows:

**Open Websocket and Request Key.** First, `pid.js` sends a request to open a Websocket in order to receive an HMAC key from the server. The server caters for such a request only once within an HTTPS session. To cater for the request, the server establishes a Websocket channel with the client, and through this channel sends a random 256-bit key  $k$ . The Websocket is subsequently closed and the rest of the communication is continued over HTTPS. Any further requests for a key in the same HTTPS session are refused by the server. If the server receives more than one request for the client, it is an indication that a malicious extension tries to impersonate the client.

**Define Mutation Observer.** The next step is to assign a *mutation observer*<sup>4</sup> to the document class. Mutation observer is a JavaScript global API that provides developers a way to react to DOM modifications. It records all the changes in the

---

<sup>4</sup><https://developer.mozilla.org/en/docs/Web/API/MutationObserver>

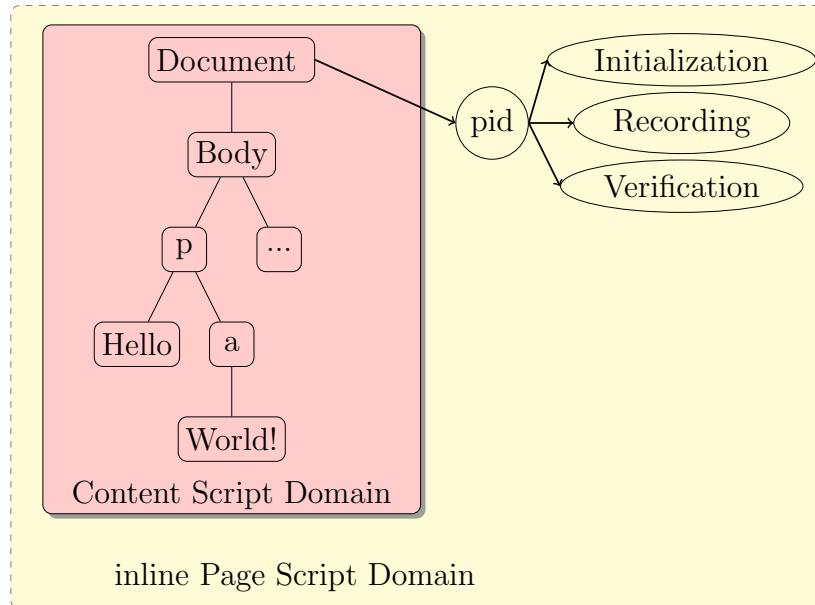


Figure 4.5: An overview of `document.pid` and the inability of extensions to modify this region of DOM.

DOM tree, including the alternations in attributes. This covers every possible DOM modification with the exception of the changes in the way events are handled in DOM. We discuss how to deal with this exception below.

**Stop Event Propagation.** In this step, `pid.js` stops assignment of new events to DOM elements by calling the `stopImmediatePropagation` method<sup>5</sup> for all elements. Note that (in DOM Level 2 and above) existing assigned events cannot be changed or removed unless the browser is presented with the reference to the registered event, and the isolated worlds principle ensures that extensions do not have access to such references.

**Define DOM Expando.** Next, the script adds an expando (i.e., an on-the-fly property) to the document node of the DOM, as shown in Figure 4.5. This property is called `document.pid`. As a property it does not change the DOM node structure, and hence is not visible to extension content scripts due to the isolated worlds principle. `document.pid` is implemented as an object with encapsulated functions. All `document.pid` functions are private (using so-called “closures”<sup>6</sup>) except for one (i.e., `document.pid.request()`) which we discuss later.

<sup>5</sup><https://developer.mozilla.org/en/docs/Web/API/Event/stopImmediatePropagation>

<sup>6</sup><https://developer.mozilla.org/en/docs/Web/JavaScript/Closures>

## Stage 2: Recording

After initialization, DOMtegrity enters a persistent passive mode and records all DOM mutations through the mutation observer. The recorded mutations include adding or removing child elements to a node, inserting or changing an attribute in a node, or modifying the data of a DOM node. The recording continues until the user’s interaction with the web page finishes and the filled form is to be posted to the server.

## Stage 3: Verification

In this stage, a page identifier (PID) containing the recorded changes in the DOM is generated. The stage starts when the function `document.pid.request()` is called. This is the only public expando function and should be called when the client “returns” the form, e.g., by clicking a “submit” button. This stage uses Web Crypto API<sup>7</sup>, a relatively new JavaScript capability to perform cryptographic operations in browser.

**Generate Page ID.** The first step is to generate the PID which consists of two parts: the list of recorded DOM mutations throughout the recording stage, and the source code of the page at the time the verification stage starts. According to the W3C standard, there are seven mutations observable. Each possible DOM mutation is encoded into a unique digit to achieve a short representation of the list. The source code (accessible to JavaScript via the `document.documentElement.innerHTML` attribute) represents the final state of the DOM elements in the page. Here we consider the protection of integrity for the whole page, but it is possible to define a custom PID to cover only part of the page. However, care must be taken to ensure that any unprotected part of the page cannot be modified by extensions to compromise security.

**Compute Assertion.** Next, a message authentication code (MAC) on the generated PID is produced in the browser using the secret key  $k$ . We opted to use HMAC with the SHA-256 hash function as our MAC. This selection is based on three reasons: first, the 128 bit security of the HMAC-SHA256 is adequate for nearly all practical web applications; second, the HMAC function is supported consistently in all modern browsers; and third, the size of the final result is always constant because of the hash (256 bits). The computed HMAC tag is sent to the server for verification as an *assertion*.

---

<sup>7</sup>[www.w3.org/TR/WebCryptoAPI](http://www.w3.org/TR/WebCryptoAPI)

**Verify Assertion.** On the server side, upon receiving the assertion, the server first checks if more than one request for fetching the HMAC key has been received earlier within the HTTPS session, and rejects the assertion if that is the case. Multiple key fetching requests indicate man-in-the-browser impersonation attacks. If only one request has been received, the server retrieves the expected the PID, computes the HMAC of the expected PID and compares it with the received assertion. Normally there is no need for the client to send the PID. The server expects no changes in the DOM other than those made by the web page scripts. Hence, the server has a specific expectation of the recorded DOM mutations and the final source code of the page, and therefore a known expected PID. The server accepts the assertion on the integrity of the page if the HMAC verification succeeds.

**Send Decision.** Depending on the decision, the server proceeds to provide or refuse further service to the client (say allowing login to online bank account). In case of refusal, the sever may communicate to the user with an error message through an out-of-band channel, e.g., sending a short text message to the user's mobile phone.

### Choosing HMAC vs. Hash

DOMtegrity uses the Websocket to securely transport a key which is later used in the generation of the HMAC tag. The Websocket channel only lasts for the duration of the key transport and is immediately closed by the server once it sends the key. An alternative approach would be to keep the Websocket open for the duration of the protocol and instead of sending an HMAC of the PID, the client can securely send a hash (say SHA-256) of the PID through the Websocket. We chose the HMAC approach to minimize the cost of communication since maintaining a full-duplex Websocket requires exchanges of ping-pong messages to keep the channel alive. By using HMAC, DOMtegrity minimizes the duration of a Websocket only for the essential purpose of transporting a short (32 bytes) key. As we will show, the computation of HMAC based on WebCryptoAPI incurs a negligible cost in the client browser. The computed HMAC tag can be sent through an XHR request over HTTPS.

#### 4.4.4 How DOMtegrity Prevents Attacks

In this section, we review a number of design choices in DOMtegrity that are essential to effectively defend against DOM manipulation attacks by malicious extensions.

**Influencing the execution of pid.js.** A malicious extension may try to influence the execution of `pid.js` through the content scripts or the injected scripts.

First of all, it cannot stop or change `pid.js` functions through its content scripts. Due to the isolated worlds principle, and that DOMtegrity procedures are defined as `document.pid` expando functions, the extension content scripts cannot block or manipulate these procedures. Furthermore, a malicious extension cannot stop or change `pid.js` functions through injection of scripts into the page. Injected scripts do not have access to the `pid.js` Websocket due to *closure*. The only interference that injected scripts can cause with DOMtegrity is to call the public function `document.pid.request()`. However, this will result in the rejection of the integrity assertion since the inject script changes DOM by adding a new `<script>` tag.

**Eavesdropping the secure channel.** The `pid.js` Websocket provides a secure communication channel between `pid.js` and the server. This channel is inaccessible to the malicious extension [46]. In other words, the extension cannot read or modify data sent through this channel.

**Polluting JavaScript variables.** A malicious extension may inject malicious scripts into the page, trying to pollute the local and global variables used by `pid.js`. First, an injected malicious script can not directly manipulate the local Websocket variable in `pid.js`. We leverage JavaScript *closure* to make a protected reference to Websocket variables to prevent any injected script from accessing it [142]. Second, an injected script cannot prevent Websocket establishment by DOMtegrity through redefining global JavaScript APIs. The isolated worlds principle prevents extensions from modifying parameters of a page’s global environment through content scripts. Hence, the only avenue to modify such global definitions would be injecting scripts into the page and making sure they run before `pid.js`, by setting `run_at` to `document_start` in the manifest. However, at `document_start` which refers to the time before the DOM is created by the browser engine, there is no DOM for the injected script to insert a `<script>` object, hence there is no influence on the parsing of `pid.js`. On the other hand, when the injected script runs after `pid.js`, DOMtegrity’s objects have already been created based on default variable definitions. If the injected script modifies the global variables, it can only affect JavaScript objects that are created after the modification. Objects that were created before the modification remain unaffected.

## 4.5 Discussion on Impersonation

The design of DOMtegrity was based on the W3C specification on “browser extensions” and the latest version is dated 23 July, 2017 [214]. We noticed that from

Table 4.2: Capabilities of extension and in-line script before and after Chrome v.58.

Capability	Extension (before Chrome v.58)	Extension (after Chrome v.58)	<code>pid.js</code>
Access the DOM	✓	✓	✓
Establish Websockets	✓	✓	✓
Block Websocket Establishment	✗	✓	✗
Block Websocket communications	✗	✗	✗
Access an expando created by <code>pid.js</code>	✗	✗	✓
Access / close Websockets established by <code>pid.js</code>	✗	✗	✓
Access / close Websockets established by the extension	✓	✓	✗

Chrome 58, launched in April, 2017, Google Chrome added a new capability, allowing the extension to block the Websocket handshake request sent from the browser to the server. This change is highlighted in Table 4.2. Discussions in the Chromium forum suggest that this change was made to prevent advertisement agencies from misusing Websockets to bypass adblocker extensions. At the time of writing this thesis, this new capability has not been included into the official W3C specification [214].

A malicious extension may try to impersonate `pid.js`, racing to engage with the server first. It can make sure to execute before `pid.js` by setting `run_at` to `document_start` in the manifest. Because of the recent change in Chrome extension API, We discuss this impersonation attack in two cases.

The first case is before Chrome v58 (which is also the latest W3C specification dated 23 July, 2017 [214]). In this case, an extension is not allowed to stop `pid.js` from sending its own Websocket request. The setting of `document_start` in the manifest of the extension enforces the execution of content scripts before parsing the loading page. However, a meaningful attack would need the user to interact with a web page that is loaded in the browser (e.g., to fill in a form or provide user credentials). The inclusion of `pid.js` before the web page HTML code ensures that the user interaction can only happen after `pid.js` sends its own Websocket establishment request. Hence, any attempt for an impersonation attack by the malicious extension is detected at the server side as a result of observing multiple Websocket establishment requests.

The second case is after Chrome v58. A new capability is added to allow the extension API to block the establishment of a Websocket by an in-line script. (We expect that this change may eventually be included into W3C). As it is stated in the webRequest API documentation (Chrome v58 and onwards) [46], the only part that an extension can influence the Websocket is blocking the handshake request sent by an in-line script. Once the Websocket channel is established by an in-line JavaScript,

an extension cannot intercept or access the data transmitted in the channel, nor can it close the channel (see Table 4.2). This change is realized in Chrome without fundamentally altering the Websocket architecture. This is because Chrome’s webRequest API can only function through the application-layer HTTP(s) based request/response pairs, and the Websocket handshake request is the only HTTP(s) based message sent in a Websocket channel. All the subsequent communication messages are transmitted in Transport-layer Websocket frames between the browser and the server, which is inaccessible to extensions. By blocking the handshake request sent at the HTTP(s) layer, Chrome effectively prevents the abuse of Websocket by an in-line JavaScript (carrying advertisements) to bypass adblocker extensions.

However, for a malicious extension to overcome DOMtegrity, it is not sufficient to just block `pid.js` from sending a handshake request, it must establish its own Websocket channel and receive the HMAC key. Calling to block `pid.js` from sending a handshake request also blocks the extension from sending its own request. In our implementation, once `pid.js` detects that sending a handshake request has been blocked by an extension, it automatically enters a loop to retry sending the request.

Hence, for a successful attack, the extension must ensure to establish a Websocket channel with server first, and through the established channel request and receive the HMAC key before it blocks `pid.js` (and the extension itself as well) from sending out any Websocket handshake request. An extension can make sure to execute before `pid.js` by setting `document_start` in the manifest. To postpone DOMtegrity’s Websocket connection, the extension may insert a large amount of JavaScript commands after the execution of its own Websocket constructor command and before the `pid.js`, hoping that this will give the extension enough time to finish its own Websocket establishment and then block DOMtegrity.

We set up an experiment to evaluate how a race condition between a malicious extension and `pid.js` operates in the browser. We injected JavaScript commands after the extension’s Websocket object creation command to delay the execution of sending the DOMtegrity Websocket handshake request in `pid.js`. Then, we gradually increased the number of JavaScript commands up to 250,000 lines of code. We measured two numbers in this experiment: first, the interval between the construction of the Websocket objects in the extension and `pid.js` and second, the interval between the dispatch of handshake requests sent by the extension and `pid.js`. The results of our evaluation are shown in Figure 4.6.

The creation of a Websocket consists of two main steps [214]: 1) constructing a new Websocket object, and 2) upon successful return of the new constructed Web-

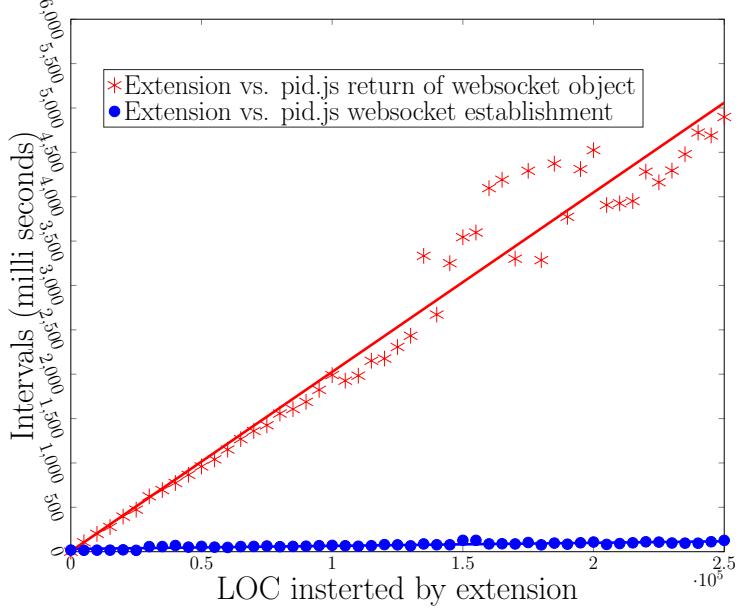


Figure 4.6: The interval between creation of Websocket objects in the extension and `pid.js` versus the interval that the opening handshake requests sent from the browser. Solid lines indicate regression of delay in each corresponding interval when extension injects commands to delay `pid.js` Websocket’s execution.

socket object, dispatching a handshake request to establish a Websocket connection. according to W3C [214], Websocket establishment run *in parallel* to the current execution context in JavaScript. In other words, Websocket handshake requests are generated *in parallel* to the JavaScript’s event loop [216], which is the execution context’s scheduler that coordinates events, user interaction, scripts, rendering, networking, and so forth in the browser. The extension can influence event loop directly by overloading the task queues but it does not have control over tasks run in parallel since they are execute by browser’s internal modules. This is evident in Figure 4.6, which shows that although the commands inserted in the extension delay the creation of the Websocket object by `pid.js`, it has little effect on postponing the dispatch of the Websocket request by `pid.js`. The minuscule increase in the delay of sending `pid.js`’s handshake request, as seen in Figure 4.6, is due to the heavy computational overhead on the CPU that has an indirect effect on the processing of the channel handshake.

Therefore, the race between the extension and `pid.js` starts exactly after the extension’s Websocket handshake request is sent. After the creation of a Websocket object in browser, the object’s “readyState” property will be initialized as “connecting” until the status of the Websocket connection is changed to either “open” in case

of successful receipt of handshake response, or “closed” otherwise. Meanwhile, there should only be one Websocket connection with “readyState” set to “connecting” at a time [97]. Thus, `pid.js`’s opening handshake request is suspended until the extension’s channel status is changed to “open”. Afterwards, the blocking process begins in the extension by firing an *onopen* event *in parallel* to `pid.js` “connecting” process. This event is queued in a task queue dedicated to tasks with the source set as “Websocket” [217]. This means the actual race is between the *onopen* event from the extension and the dispatch of `pid.js` Websocket opening handshake request.

We measured the time that blockage happened after receiving of the extension’s opening handshake response and compared it with the time that `pid.js` opening handshake is dispatched from browser after extension’s channel is open. The results are shown in Figure 4.7. From the results, we can draw two conclusions. First, the time it takes for the extension to block the Websocket is always longer than that needed by `pid.js` to send the opening handshake request. This is because the Websocket’s opening handshake request generation consists of a fixed number of steps without depending on other elements in the web page [215], while the blocking process involves computationally more intensive webRequest API calls and transmission of messages between the content script and the background script. Second, when more commands are inserted by the extension, it affects more on the blocking process than on the sending of the handshake request, as evident by the steeper line for the completion of blocking in the extension. This is because all of the steps in sending the handshake request execute *in parallel* to normal execution of commands in the JavaScript execution context. The tasks in the blocking process run in the event loop and the corresponding task queues. They are dependant on the complexity of other elements in the web page. Hence, overflowing the task queues by inserting excessive commands would delay the blocking process more than it does on the generation of `pid.js`’s Websocket handshake request.

In conclusion, we have shown, both empirically and analytically, that the new capability of the Chrome WebExtension network access does not prevent the working of DOMtegrity. A malicious extension is still unable to impersonate `pid.js`.

## 4.6 Implementation and Evaluation

In this section, we describe how we implemented a number of proof-of-concept malicious extensions to test our solution in several attack scenarios and provide performance measurements.

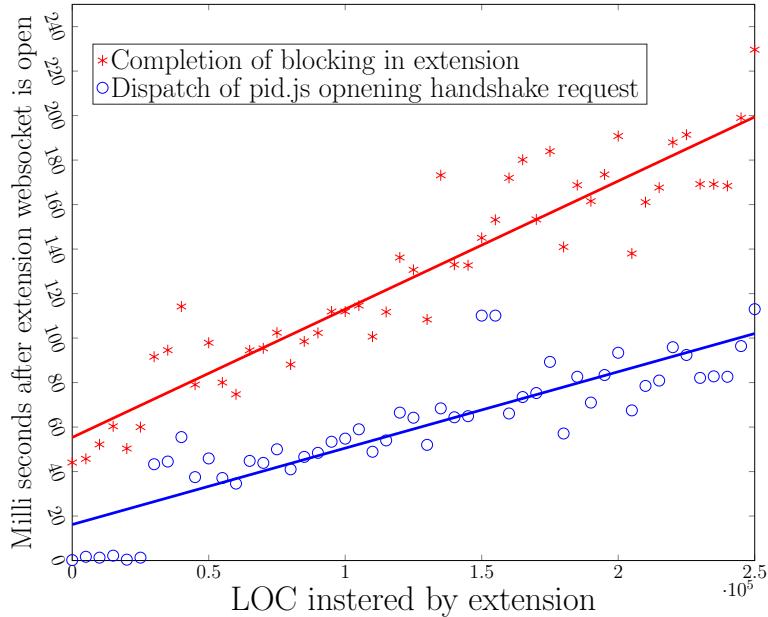


Figure 4.7: The time when `pid.js`'s opening handshake dispatch versus when blocking executed in the extension. Both numbers are measured once malicious extension's Websocket is opened successfully. Solid lines indicate linear regression of in each series when extension injects commands to delay `pid.js` Websocket's execution.

On the client side, DOMtegrity is implemented as a single JavaScript program which is integrated in-line within a `<script>` tag in the beginning of a web page. On the server side, we implemented the server using Node.js version 4.4.0. All cryptographic operations in `pid.js` are programmed as asynchronous operations using JavaScript *Promise* objects<sup>8</sup>.

#### 4.6.1 Confirming DOMtegrity Effectiveness

**Detecting Online Banking Attacks.** To confirm that our implementation of DOMtegrity can detect the attacks we discussed in Section 4.3, we implemented copies of the online banking web pages for both systems on our local server and embedded `pid.js` in-line. Then, we re-ran the attacks by the malicious extensions we developed on Chrome and Firefox. In both cases the server was able to successfully detect the malicious modifications made on the web pages and block further requests from the client.

**Detecting Other Possible DOM Modifications.** To confirm that our implementation of DOMtegrity can detect other possible DOM modifications, we consid-

---

<sup>8</sup>[https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global\\_Objects/Promise](https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_Objects/Promise)

ered a comprehensive list of changes extensions can make to DOM and developed extensions that make such changes through content scripts. These changes include:

1. insert a new DOM element into the tree;
2. remove a targeted DOM element from the tree;
3. hide a targeted DOM element and replace it with its own element (possibly of an identical type) with a different ID;
4. change the style of a targeted DOM element; and
5. embed another script file which in turn changes an attribute of a targeted DOM element.

We developed five extensions (based on WebExtensions), each making one of the above modifications. All these extensions are tested on a simple login web page, which contains username and password text boxes and a “Sign in” button, with `pid.js` embedded in-line. We tested each of our extensions on Chrome and Firefox. As we expected, in all the experiments our server was able to detect the malicious DOM modifications on the client side.

#### 4.6.2 Performance Evaluations

On the client side, the web page is run in Firefox v50.1 and Chrome v54 on a machine equipped with Intel Core i7 2.8 GHz with 8 GB of RAM and Windows 7 Enterprise. The server is set up on a machine with windows 8.1 x64 Enterprise Edition equipped with Intel Core i5 2.3 GHz with 8 GB of RAM.

**File Size.** The client side JavaScript is 550 lines of code and adds 21.6 KB in the normal mode and 6.33 KB in the minified mode to the original web page source code. Our simple login page, the HSBC web page and the Barclays web page are 31.5 KB, 2.1 MB and 3.6 MB, respectively. The overhead of the DOMtegrity client source code is relatively small compared to those of other popular JavaScript frameworks. For example, the popular JQuery framework<sup>9</sup> adds 84.6 KB to the web page in the minified mode. The server side Node.js implementation is 240 lines of code with a size of 4.25 KB.

**Computation load.** The computation load of the initialization stage is proportional to the number of elements in the web page since the browser needs to stop

---

<sup>9</sup><https://jquery.com>

Table 4.3: Average elapsed times for stopping event propagation in Chrome and Firefox for our experimental web pages

	#Elements	Total time (ms)		Time/Element (ms)	
		Chrome	Firefox	Chrome	Firefox
Simple login page	22	16.53	15.64	<b>0.75</b>	<b>0.71</b>
Simulated HSBC page	987	713.68	485.08	<b>0.72</b>	<b>0.49</b>
Simulated Barclays page	1283	839.83	624.76	<b>0.65</b>	<b>0.49</b>

event registration for every node of the DOM. We measured the time it takes for this step to complete for our own login page and for the comparatively richer HSBC and Barclays online banking pages. For each page we ran the experiment 100 times and we report the average here. For our login page, this step took 15.64 ms on Firefox and 16.53 ms on Chrome to complete, resulting in an average of 0.71 to 0.75 ms per DOM element. For the Barclays page, the richest page, this step took 624.76 ms on Firefox and 839.83 ms on Chrome to complete, resulting in an average of 0.49 to 0.65 ms per DOM element. Further details are reported in Table 4.3.

The recording stage only stores an encoding of the DOM change for every DOM modification and incurs a negligible computational overhead. In our experiments, the latency for recording each mutation is 0.005 ms.

The verification stage requires the calculation of PID and HMAC tag. In our measurements, the average elapsed time for computation of PID is 1.97 ms in Chrome and Opera, and 2.79 ms in Firefox, and the average elapsed time for computing the HMAC tag is 2.63 ms in Chrome and Opera, and 2.68 ms in Firefox. The box plots of elapsed times for 100 executions in Firefox and Chrome are illustrated in Figure 4.8. All values are rounded up to the closest 0.01 ms.

Computations on the server side are very efficient. The most time consuming step on the server side is retrieving PID from storage which takes 1.96 ms on average. It takes 0.17 ms to compute a HMAC tag and another 0.03 ms to compare the tag against the received. The average elapsed time for 100 executions of each step on the server side is shown in Table 4.4. All values are rounded up to the closest 0.01 ms.

**Communication Bandwidth.** DOMtegrity is designed to be efficient in terms of required communication bandwidth. The key and the MAC tag are only 32 bytes each, amounting to a negligible fraction of the usual data transmission between the client and the server. The embedded JavaScript code is relatively compact (21.6 KB in the

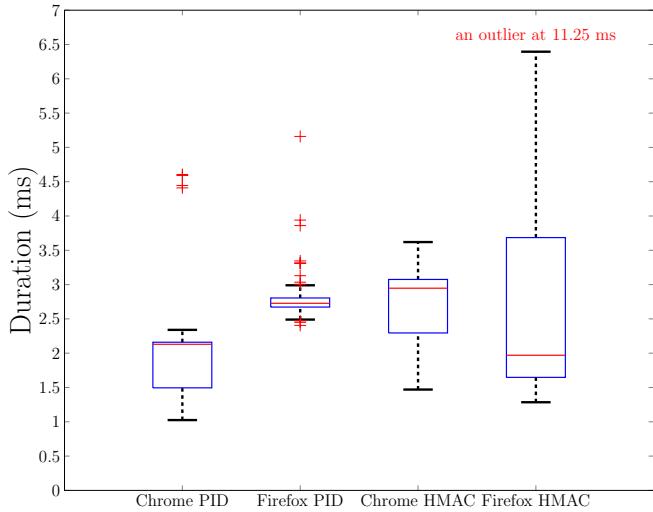


Figure 4.8: Box plots of elapsed times for PID and HMAC calculations in 100 executions in Chrome and Firefox.

Table 4.4: Average and standard deviation of the elapsed times on the server side for 100 executions of each step of the protocol

Step	Average time (ms)	STD (ms)
Key generation	0.02	0.02
PID retrieval	1.96	1.59
HMAC calculation	0.17	0.01
Decision	0.03	0.02

normal and 6.33 KB in the minified mode), as compared to other popular JavaScript frameworks such as JQuery (84.6 KB in the minified mode). The establishment of the Websocket is also efficient as the underlying technology is designed to be lightweight. By the design of DOMtegrity, the duration of the Websocket channel is kept to the minimum only for the essential purpose of transporting the HMAC key.

#### 4.6.3 Compatibility with Real-world Extensions

DOMtegrity is designed to detect all DOM changes. The strict policy is to reject any detected DOM modification that the server does not expect. However, such a strict policy is obviously in contradiction with existing extensions that work by modifying the DOM. Examples of such extensions include Grammarly (a popular grammar and spell checker) and LastPass (a popular password manager). In this section, we investigate the compatibility of DOMtegrity with *real-world* extensions.

**Real-world extension set.** For this experiment, we have downloaded a large set

of extensions from the Chrome Web Store and the official Mozilla Add-on repositories. Overall, we investigated more than 14,000 WebExtensions-based extensions in the two repositories, as follows:

- all extensions from Chrome’s Starter Kit list,
- all extensions from Chrome’s Editor Picks list,
- all extensions returned with the search keyword “block”,
- all extensions returned with the search keyword “blocker”,
- all extensions with more than 100 active users in each Chrome Web Store extension category, and
- all WebExtension-based add-ons in Mozilla’s top 1,000 most popular extensions (57 extensions).

We installed each extension in a mint instance of the browser, then we requested a DOMtegrity-protected web page, i.e., a page in which the `pid.js` script was embedded. When the page was completely loaded in the browser, we recorded the generated PID in the presence of the extension on the client side, plus the assertion verification result on the server side.

**Results.** We compared the generated PID on the client side with the expected PID on the server side for each rejected extension in order to investigate the type of modification they applied. The W3C specification on DOM categorizes page mutations into three groups: *attributes*, *characterData* and *childList* [212]. The *attributes* category includes mutations involving modifications of attributes of existing nodes. *CharacterData* refers to mutations that change any data between the opening and closing tags of a text node. Finally, *ChildList* includes mutations that involve insertion or removal of nodes in the DOM tree. We investigated the generated PID on the client side and classified the rejected extensions into the above categories. A rejection by the server may be caused by a mixture of the mutation types. In that case, the PID records every type of the mutations.

Overall, 15% of the extensions caused rejection of the assertion. This shows that DOMtegrity is compatible with 85% of the wide range of extensions we tested. Among the 15% rejections, 86% of them involved attribute mutations, 2% characterData mutations, and 98% childList mutations.

Table 4.5: Likelihood of mutation type occurrence among tested rejected extensions for each mutation type

Mutation Type	Occurrence Probability
attributes	43.9%
characterData	0.2%
childList	55.9%

On the other hand, if we simply record every mutation caused by the extension in the PID, the probability of occurrence for each of mutations types for attribute, characterData and childList mutations was 43.9%, 0.2% and 55.9% respectively, as shown in Table 4.5.

**Possible mitigations.** One possible mitigation strategy to accommodate extensions that are incompatible with the rather strict ‘no mutations allowed’ policy is to consider a more flexible policy that allows certain specific DOM modifications. Our protocol needs to be slightly modified to enable such flexible policies. Since it is difficult for the server to predict modifications on the client side, `pid.js` will need to send the PID to the server along with the assertion. The PID consists of the recorded mutations and the final source code. The server can then check the PID against a set of policies to decide if the mutations are acceptable.

Note that our attacks on online banking systems lie in the CharacterData category since we changed the fields within a text node. Recall that only 47 extensions (0.3% of whole data set) and more importantly, 0.2% of the recorded mutations by extensions fall within this category based on an extensive analysis of 14k extensions. Hence, a simple policy that does not allow CharacterData mutations can effectively detect such attacks through DOMtegrity, while remaining overwhelmingly compatible with the widely used extensions.

Overall, further compatibility can be gained by modifying the protocol resulting in the client sending more data (i.e., the PID) and the server performing slightly more complex verification.

## 4.7 Further Discussion

**Browser Parsing Inconsistencies.** In practice, given the same web page source code, different browsers may parse it differently. In some circumstances, a browser might change the source code. For example, during the testing of our protocol, DOMtegrity identified two particular changes, as shown in Figure 4.9. These changes

<pre>&lt;input class="action-button next"       id="buttonTest" type="button"       value="Sign in"&gt;</pre>	$\rightarrow$ <pre>&lt;input id="buttonTest"       class="action-button next"       value="Sign in" type="button"&gt;</pre>
(a) Original source code in Firefox	(b) Parsed source code in Firefox
<pre>&lt;li&gt;   &lt;li&gt;     &lt;img src="img/Money.png"&gt;     &lt;div&gt;       &lt;h3&gt;Looking after your money&lt;/h3&gt;       &lt;p&gt;Get&lt;/p&gt;       &lt;a href="https://...&gt;Managing&lt;/a&gt;     &lt;/div&gt;   &lt;/li&gt; &lt;/li&gt;</pre>	$\rightarrow$ <pre>&lt;li&gt;&lt;/li&gt; &lt;li&gt;   &lt;img src="img/Money.png"&gt;   &lt;div&gt;     &lt;h3&gt;Looking after your money&lt;/h3&gt;     &lt;p&gt;Get&lt;/p&gt;     &lt;a href="https://...&gt;Managing&lt;/a&gt;   &lt;/div&gt; &lt;/li&gt;</pre>
(c) Original source code in Chrome	(d) Parsed source code in Chrome

Figure 4.9: Examples of source code modifications during parsing in browsers

are inconsistent between browsers, appear unnecessary and are not widely documented. They do not alter the content of the page, but they modify the DOM structure.

Fortunately, such changes are rare, and can be conveniently addressed by testing. We recommend that before integrating DOMtegrity into a web page, the page is tested against targeted browsers to identify if there is any modification to the source code made by the browser. In case that modifications are observed, the web page can be rectified accordingly to avoid those changes. This will ensure that in the field deployment, DOMtegrity will only catch modifications to DOM made by malicious extensions, not by the browser itself. As a longer-term solution, we recommend browser vendors to refrain making ad hoc changes to the original HMTL source code, and adhere to the W3C specifications [213] in parsing and construction of the DOM<sup>10</sup>.

**Dynamic Web Pages.** A dynamic web page is one with variable content depending on the user or her actions. This is done by either server-side or client-side scripting, or a mixture of both.

If only server-side scripting is used, a web page is constructed on the server side at the time of request and transmitted to the client. No further changes to the DOM are expected in this case. Hence, such pages can be protected using DOMtegrity as it is designed.

If client-side scripting is used, the dynamic web page DOM is modified in-browser based on the user's interactions with the page. In this case, there would be no way for

---

<sup>10</sup>Based on private communication with a contact in W3C/Google, the subtle parsing inconsistencies between browsers, which are caught by DOMtegrity in Figure 4.9, are most likely implementation bugs in the browsers and should be fixed in future releases.

the server to predict user’s interactions with the page and hence it would be necessary for `pid.js` to send the PID along with the HMAC tag to the server so that a decision on the integrity of the page can be made based on the server’s policies.

**Private Mode.** Extension availability policies in private mode are different across browsers. Firefox permits extensions to function in private mode. In contrast, Chrome disables the extensions by default in its private mode (incognito). In each case, DOMtegrity functions as normal, regardless if the extensions are enabled in the client browser.

**Content Security Policy (CSP).** CSP (Level 3) is a W3C working draft<sup>11</sup> with the goal of preventing cross-site scripting attacks. Based on the current version of this draft, browser extensions are not prevented from altering the CSP defined by the web page. Hence, extensions can prevent the execution of the web page JavaScript code altogether by setting `script-src` to `none` in the CSP response header. This would effectively disable DOMtegrity, and in fact disable any web page with embedded JavaScript code. However, the extension’s ability to alter CSP is in conflict with the fact that CSP is designed to be set by the server. The critical assumption made in the current working draft on CSP is that extensions are regarded as “fully trusted”. However, in light of the threats of malicious extensions demonstrated in this section and many previous works [173, 2], we urge W3C to take into account potentially malicious behaviour of extensions in the revision of the CSP draft.

**Confidentiality.** Although DOMtegrity guarantees end-to-end integrity, it is not designed to provide any confidentiality against malicious extensions. Through their access to DOM and HTTP(S) traffic data, extensions are able to read both the server and user-provided contents of the web page.

**Implementation Limitations.** DOMtegrity has been successfully implemented and evaluated in an experimental setting; however, one should include more possible attack scenarios in the real-world application. Specifically, checking of the double receipt of assertion through the same HTTPS session is prone to Denial of Service (DOS) attack by a malicious extension. This way, DOMtegrity server could stop providing service and consequently, cease `pid.js` to function. This implementation weakness is yet to be resolved in the future developments of our project.

---

<sup>11</sup>[www.w3.org/TR/CSP3](http://www.w3.org/TR/CSP3)

## 4.8 Industry Acknowledgement

We have been in contact with professionals from browser vendors in order to have a better insight on the extensions and JavaScript execution mechanisms. We specifically would like to thank Tobie Langel, web standards and open source consultant in Google and W3C, and Jake Archibald, developer advocate for Google Chrome. Their recommendations were crucial in terms of understanding the implementation aspects of W3C standards, WebExtensions, event loops, task sources and webSocket execution in JavaScript.

## 4.9 Summary

In this chapter, we presented DOMtegrity, a light-weight JavaScript based solution to provide end-to-end protection of integrity for web content from the point of delivery at a sever to the final rendering in a client’s browser. Our solution works with the standard WebExtensions framework and does not require modifying existing architectures of web browsers, nor using any external hardware device. As part of the evaluation, we implement two attacks on real-world online banking websites: HBSC and Barclays, to demonstrate how malicious extensions can compromise the online banking security, and how DOMtegrity can effectively prevent such attacks as well as other man-in-the-browser attacks caused by malicious extensions. We run an extensive study of the top 14k extensions to investigate the prevalence and types of DOM changes and confirm that DOMtegrity is compatible with the majority of widely-used extensions. We present detailed timing measurements to show that DOMtegrity is efficient and adds only a relatively small overhead to the performance on both the client and the server sides.

# Chapter 5

## Conclusion and Future Work

### 5.1 Summary

In this thesis, we introduced two novel approaches in tamper-evident technologies. On physical tamper-evidence, we developed a new technique to determine the authenticity of a paper sheet based on its texture patterns. Moreover, on cyber tamper-evidence, we proposed a JavaScript based protocol to securely detect malicious modifications by browser extensions.

Chapter 2 introduced the state-of-the-art trends in tamper evident solutions. We surveyed different approaches in physical and cyber detection of tampering. Then, we demonstrated the challenges in each approach and discussed how it would influence future developments.

In Chapter 3, we presented “paper fingerprinting” as a case study for a physical tamper evident solution. Our proposal was based on the idea that a paper sheet texture contains random patterns that could be utilized as its fingerprint. In other words, a paper sheet is a Physical Unclonable Function (PUF) and thus, it can not be physically cloned by adversaries. Previous research in the field was based on reflection of laser beam or light from the paper sheet’s surface. However, we argued that the paper sheet texture contains more entropy than its surface. Hence, we proposed to fingerprint a paper sheet based on measuring the translucent patters when light transmits through the paper sheet instead of measuring the light reflections from its surface.

We implemented our fingerprinting mechanism with an off-the-shelf digital camera and a commodity light source. Unlike other research in the literature, our method required merely a single shot of the texture patterns when the paper is back-lit by the light source. The captured sample is then inspected by texture analysis algorithms to

generate a 2048 bit identifier, also known as the *paper fingerprint*. The paper fingerprint can withstand various rough handling situations with perfect (100%) acceptance rate. Furthermore, the 807 bits entropy make it suitable for large-scale applications.

Finally, we proposed two authentication approaches to verify the authenticity of a paper sheet. Using our method, illegal clone of a legal document would be detected in the authentication process because an attacker could not imitate the underlying texture of the paper sheet.

Furthermore, in Chapter 4, we proposed a lightweight JavaScript framework to detect clandestine modifications to the web page’s DOM tree by malicious browser extensions. To demonstrate the threat, we implemented two real-world attacks on HSBC and Barclays on-line banking websites. We demonstrated that a malicious extension was able to access the victim’s banking account and transfer funds to the adversary’s account without the user noticing it.

Then, we designed *DOMtegrity* as a protocol to detect malicious DOM modifications, a.k.a. mutations, when the web page is being parsed in the client’s browser. In our analysis, we compared the executions of an extension with in-line JavaScript code and discovered in two major differences: first, the limitation of extension’s access to a Websocket channel and second, extension’s inability to detect dynamic on-the-fly extended DOM properties (also known as DOM expando). We leveraged these differences to design our protocol.

The client side implementation of DOMtegrity is embedded into the web page source code as an in-line script called `pid.js`. It records all the DOM mutations from the time page starts loading. Later, it securely sends the recorded DOM mutations and the page’s source code, the combination that we called *Page IDentifier (PID)* to the server. Finally, DOMtegrity server decides about the received information by comparing it against their expected PID. The server’s decision leads to rejection or acceptance of the upcoming requests through the channel with the same TLS session ID.

We utilized modern HTML5 APIs such as WebSockets, WebCrypto and MutationObserver in order to effectively implement our protocol. All mainstream modern browsers support the above mentioned APIs. We have evaluated our protocol in various attack scenarios.

The main advantage of DOMtegrity is that it protects users from a malicious extension’s modifications on the DOM, without requiring changing the browser’s existing internal design or extra hardware token. Our solution can be easily deployed by embedding in-line JavaScript code in the web page.

## 5.2 Future Work

Future work is suggested as follows:

- In Chapter 3, we focused on evaluating our method on regular A4 paper sheets. In future, we plan to extend our research to other types of paper, such as thermal paper, labels and other thicker forms of paper as long as the light can transmit through. However, it is likely that adjustments in the intensity of the light, camera settings, Gabor filter scale and orientation will be required due to the differences in the thickness and the materials of the paper. These questions will be addressed in the future developments of our research.
- In Chapter 3, we would like to the application of the proposed paper fingerprinting technique to prevent counterfeiting of banknotes, passports and other legal documents. Consequently, it has the potential to be used as a new approach in fighting against forgery.
- In Chapter 4, we have evaluated our protocol against static web-pages. However, this might be not applicable to more sophisticated web-pages that currently exist in the internet. Thus, we would like to investigate extending DOMtegrity to more dynamic web pages.
- In Chapter 4, we suggest two approaches to develop DOMtegrity in more real-world applications. First, we need to evaluate more modern web technologies (such as single page frameworks like angular.js, react and etc.) and second, we should examine the effects of the concurrent execution of `pid.js` and other JavaScript frameworks (such as jQuery, bootstrap and etc.).
- Finally, in Chapter 4, we would like to establish a more inclusive server-side decision engine by analysing the real-world extension behaviour. The comprehensive examination of the legitimate DOM mutations by extensions would aid DOMtegrity server to detect malicious web page modifications more accurately.

# Bibliography

- [1] Dennis G. Abraham, George M. Dolan, Glen P. Double, and James V. Stevens. Transaction security system. *IBM Systems Journal*, 30(2):206–229, 1991.
- [2] Manal Adham, Amir Azodi, Yvo Desmedt, and Ioannis Karaolis. How to attack two-factor authentication internet banking. In *International Conference on Financial Cryptography and Data Security*, pages 322–328. Springer, 2013.
- [3] Miguel Alcaraz-Rivera, J Javier Báez-Rojas, and Kang Der-Kuan. Development of a fully functioning digital hologram system. In *Integrated Optoelectronic Devices 2008*, pages 69120S–69120S. International Society for Optics and Photonics, 2008.
- [4] Wade Alcorn, Christian Frichot, and Michele Orru. *The Browser Hacker’s Handbook*. John Wiley & Sons, 2014.
- [5] Ross J Anderson. *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons, 2010.
- [6] William A Arbaugh, David J Farber, and Jonathan M Smith. A secure and reliable bootstrap architecture. In *Security and Privacy, 1997. Proceedings., 1997 IEEE Symposium on*, pages 65–71. IEEE, 1997.
- [7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.
- [8] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Francois-Xavier Standaert, and Christian Wachsmann. A formalization of the security features of physical functions. In *Security and Privacy (SP), 2011 IEEE Symposium on*, pages 397–412. IEEE, 2011.

- [9] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. Memory leakage-resilient encryption based on physically unclonable functions. In *Towards Hardware-Intrinsic Security*, pages 135–164. Springer, 2010.
- [10] Frederik Armknecht, Daisuke Moriyama, Ahmad-Reza Sadeghi, and Moti Yung. Towards a unified security model for physically unclonable functions. In *Cryptographers Track at the RSA Conference*, pages 271–287. Springer, 2016.
- [11] David Aucsmith. Tamper resistant software: An implementation. In *Information Hiding*, pages 317–333. Springer, 1996.
- [12] Sruthi Bandhakavi, Samuel T King, Parthasarathy Madhusudan, and Marianne Winslett. Vex: Vetting browser extensions for security vulnerabilities. In *USENIX Security Symposium*, volume 10, pages 339–354, 2010.
- [13] Bank of England. Counterfeit bank of england banknotes. <http://www.bankofengland.co.uk/banknotes/Pages/about/counterfeits.aspx>. Accessed: 2017-10-13.
- [14] Barclays Corporate. Roadmap to psd2. We Link available at [https://www.barclayscorporate.com/content/dam/corppublic/corporate/Documents/Operating\\_Internationally/roadmap-to-PSD2-infographic.pdf](https://www.barclayscorporate.com/content/dam/corppublic/corporate/Documents/Operating_Internationally/roadmap-to-PSD2-infographic.pdf) – accessed in 2018.
- [15] D.W. Bauder. An anti-counterfeiting concept for currency systems. Technical report, Sandia National Labs, Albuquerque, NM, 1983. Technical Report PTK-11990.
- [16] DW Bauder. An anti-counterfeiting concept for currency systems. *Sandia National Labs, Albuquerque, NM, Tech. Rep. PTK-11990*, 1983.
- [17] Fokko Beekhof, Sviatoslav Voloshynovskiy, Oleksiy Koval, Renato Villán, and Thierry Pun. Secure surface identification codes. In *Electronic Imaging 2008*, pages 68190D–68190D. International Society for Optics and Photonics, 2008.
- [18] Charles H Bennett, Gilles Brassard, Seth Breidbart, and Stephen Wiesner. Quantum cryptography, or unforgeable subway tokens. In *Advances in Cryptology*, pages 267–275. Springer, 1983.
- [19] J Blackledge. Making money from fractals and chaos: Microbar. *Mathematics Today*, 35(6):170–173, 1999.

- [20] Manuel Blum. Program checking. In *Foundations of Software Technology and Theoretical Computer Science*, pages 1–9. Springer, 1991.
- [21] Manuel Blum. Program result checking: A new approach to making programs more reliable. *Automata, Languages and Programming*, pages 1–14, 1993.
- [22] Manuel Blum and Sampath Kannan. Designing programs that check their work. *Journal of the ACM (JACM)*, 42(1):269–291, 1995.
- [23] Tyler Blumenthal, Jeevan Meruga, P Stanley May, Jon Kellar, William Cross, Krishnamraju Ankireddy, Swathi Vunnam, and QuocAnh N Luu. Patterned direct-write and screen-printing of nir-to-visible upconverting inks for security applications. *Nanotechnology*, 23(18):185305, 2012.
- [24] Christoph Bösch, Jorge Guajardo, Ahmad-Reza Sadeghi, Jamshid Shokrollahi, and Pim Tuyls. Efficient helper data key extractor on fpgas. *Cryptographic Hardware and Embedded Systems—CHES 2008*, pages 181–197, 2008.
- [25] British Broadcasting Channel (BBC). Fake passport seizures at uk borders at five-year high. <http://www.bbc.co.uk/news/uk-england-35959213>. Accessed: 2017-10-13.
- [26] Jorgen Brosow and Erik Furugard. Method and a system for verifying authenticity safe against forgery, August 19 1980. US Patent 4,218,674.
- [27] Benedict J Brown, Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, and Tim Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling theran wall paintings. In *ACM Transactions on Graphics (TOG)*, volume 27, page 84. ACM, 2008.
- [28] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.
- [29] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In *CRYPTO*, volume 6841, pages 51–70. Springer, 2011.
- [30] BSA The Software Alliance. Bsa global software survey. <http://globalstudy.bsa.org/2016/>. Accessed: 2017-10-16.

- [31] James DR Buchanan, Russell P Cowburn, Ana-Vanessa Jausovec, Dorothee Petit, Peter Seem, Gang Xiong, Del Atkinson, Kate Fenton, Dan A Allwood, and Matthew T Bryan. Forgery: Fingerprinting documents and packaging. *Nature*, 436(28):475, 2005.
- [32] James DR Buchanan, Russell P Cowburn, Ana-Vanessa Jausovec, Dorothée Petit, Peter Seem, Gang Xiong, Del Atkinson, Kate Fenton, Dan A Allwood, and Matthew T Bryan. Forgery:fingerprintingdocuments and packaging. *Nature*, 436(7050):475–475, 2005.
- [33] Philippe Bulens, F-X Standaert, and J-J Quisquater. How to strongly link data and its medium: the paper case. *IET Information Security*, 4(3):125–136, 2010.
- [34] Philippe Bulens, F-X Standaert, and J-J Quisquater. How to strongly link data and its medium: the paper case. *IET Information Security*, 4(3):125–136, 2010.
- [35] Heike Busch, Stefan Katzenbeisser, and Paul Baecher. Puf-based authentication protocols—revisited. In *International Workshop on Information Security Applications*, pages 296–308. Springer, 2009.
- [36] Jan Cappaert, Nessim Kissnerli, Dries Schellekens, and Bart Preneel. Self-encrypting code to protect against analysis and tampering. In *1st Benelux Workshop Inf. Syst. Security*, 2006.
- [37] Jan Cappaert, Bart Preneel, Bertrand Anckaert, Matias Madou, and Koen De Bosschere. Towards tamper resistant code encryption: Practice and experience. *Information Security Practice and Experience*, pages 86–100, 2008.
- [38] Jan Cappaert, Bart Preneel, Bertrand Anckaert, Matias Madou, and Koen De Bosschere. Towards tamper resistant code encryption: Practice and experience. *Information Security Practice and Experience*, pages 86–100, 2008.
- [39] Lisa J Carnahan and Miles E Smid. Security requirements for cryptographic modules. Technical report, 1994.
- [40] Hoi Chang and Mikhail J Atallah. Protecting software code by guards. In *Digital Rights Management Workshop*, volume 2320, pages 160–175. Springer, 2001.

- [41] Abbas Cheddad, Joan Condell, Kevin Curran, and Paul Mc Kevitt. A secure and improved self-embedding algorithm to combat digital document forgery. *Signal Processing*, 89(12):2324–2332, 2009.
- [42] Yuqun Chen, M Kivanç Mihçak, and Darko Kirovski. Certifying authenticity via fiber-infused paper. *ACM SIGecom Exchanges*, 5(3):29–37, 2005.
- [43] Yuqun Chen, M Kivanç Mihçak, and Darko Kirovski. Certifying authenticity via fiber-infused paper. *ACM SIGecom Exchanges*, 5(3):29–37, 2005.
- [44] Yuqun Chen, Ramarathnam Venkatesan, Matthew Cary, Ruoming Pang, Saurabh Sinha, and Mariusz H Jakubowski. Oblivious hashing: A stealthy software integrity verification primitive. In *International Workshop on Information Hiding*, pages 400–414. Springer, 2002.
- [45] Tom Chothia and Vitaliy Smirnov. A traceability attack against e-passports. In *Financial Cryptography and Data Security*, pages 20–34. Springer, 2010.
- [46] Chrome Developers. Webrequest api. Web Link available at <https://developer.chrome.com/extensions/webRequest> – accessed in 2017, September 2017.
- [47] GB Clarke, D Van Dijk, and SM Devadas. Controlled physical random functions. *Proceedings. 18th Annual*, pages 149–160, 2002.
- [48] William Clarkson, Tim Weyrich, Adam Finkelstein, Nadia Heninger, J Alex Halderman, and Edward W Felten. Fingerprinting blank paper using commodity scanners. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 301–314. IEEE, 2009.
- [49] Richard Clayton. Techno-risk. Cambridge International Symposium on Economic Crime, 2003. <https://www.cl.cam.ac.uk/~rnc1/talks/030910-TechnoRisk.pdf>.
- [50] Christian Collberg, Clark Thomborson, and Douglas Low. A taxonomy of obfuscating transformations. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 1997.
- [51] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation-tools for software protection. *IEEE Transactions on software engineering*, 28(8):735–746, 2002.

- [52] Victor V Cordell, Nittaya Wongtada, and Robert L Kieschnick Jr. Counterfeit purchase intentions: role of lawfulness attitudes and product traits as determinants. *Journal of Business Research*, 35(1):41–53, 1996.
- [53] National Research Council et al. *Counterfeit deterrent features for the next-generation currency design*, volume 472. National Academies Press, 1993.
- [54] Crime in England and Wales. Crime in england and wales: year ending sept 2016. <https://www.ncsc.gov.uk/news/crime-survey-england-and-wales-includes-cyber-crime-statistics-first-time>. Accessed: 2017-10-13.
- [55] John Daugman. High confidence visual recognition of persons by a test of statistical independence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(11):1148–1161, Nov 1993.
- [56] John Daugman. Biometric decision landscapes. Technical Report UCAM-CL-TR-482, University of Cambridge, 2000.
- [57] John Daugman. The importance of being random: statistical principles of iris recognition. *Pattern Recognition*, 36(2):279 – 291, 2003. Biometrics.
- [58] John Daugman. How iris recognition works. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):21–30, 2004.
- [59] John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.
- [60] Gerald DeJean and Darko Kirovski. Rf-dna: Radio-frequency certificates of authenticity. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 346–363. Springer, 2007.
- [61] Gerald DeJean and Darko Kirovski. Rf-dna: Radio-frequency certificates of authenticity. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 346–363. Springer, 2007.
- [62] Mohan Dhawan and Vinod Ganapathy. Analyzing information flow in javascript-based browser extensions. In *Computer Security Applications Conference, 2009. ACSAC'09. Annual*, pages 382–391. IEEE, 2009.

- [63] Saar Drimer, Steven J Murdoch, and Ross Anderson. Optimised to fail: Card readers for online banking. In *International Conference on Financial Cryptography and Data Security*, pages 184–200. Springer, 2009.
- [64] Carl Ellison. Tamper resistant method and apparatus, June 6 2000. US Patent 6,073,237.
- [65] Funda Ergün, Sampath Kannan, S Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 259–268. ACM, 1998.
- [66] Gary L Friedman. The trustworthy digital camera: Restoring credibility to the photographic image. *IEEE Transactions on consumer electronics*, 39(4):905–910, 1993.
- [67] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 148–160. ACM, 2002.
- [68] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience*, 16(11):1077–1098, 2004.
- [69] Blaise Laurent Patrick Gassend. *Physical random functions*. PhD thesis, Massachusetts Institute of Technology, 2003.
- [70] Samuel Gibbs. Google pulls malware Twitter and Feedly extensions for Chrome. The Guardian, 20 Jan. 2014, <http://www.theguardian.com/technology/2014/jan/20/google-malware-twitter-feedly-chrome-browser> (accessed in 2016).
- [71] Vladimir I Girnyk, Igor V Tverdokhleb, and Andrey A Ivanovsky. Combined optical/digital security devices. In *Electronic Imaging*, pages 322–328. International Society for Optics and Photonics, 2000.
- [72] Hari Goyal. Properties of paper. Web page available at <http://www.paperonweb.com/paperpro.htm> – accessed in 2015, 2015.
- [73] Sidney N Graybeal and Patricia Bliss McFate. Getting out of the starting block. *Scientific American*, 261(6):61–67, 1989.

- [74] Daniel Gruhl and Walter Bender. Information hiding to foil the casual counterfeiter. In *International Workshop on Information Hiding*, pages 1–15. Springer, 1998.
- [75] Jorge Guajardo, Sandeep S Kumar, Geert Jan Schrijen, and Pim Tuyls. Fpga intrinsic pufs and their use for ip protection. In *CHES*, volume 4727, pages 63–80. Springer, 2007.
- [76] Jorge Guajardo, Sandeep S Kumar, Geert Jan Schrijen, and Pim Tuyls. Fpga intrinsic pufs and their use for ip protection. In *CHES*, volume 4727, pages 63–80. Springer, 2007.
- [77] Jorge Guajardo, Boris Škorić, Pim Tuyls, Sandeep S Kumar, Thijs Bel, Antoon HM Blom, and Geert-Jan Schrijen. Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions. *Information Systems Frontiers*, 11(1):19–41, 2009.
- [78] Arjun Guha, Matthew Fredrikson, Benjamin Livshits, and Nikhil Swamy. Verified security for browser extensions. In *2011 IEEE Symposium on Security and Privacy*, pages 115–130. IEEE, 2011.
- [79] Jing-Ming Guo, Soo-Chang Pei, and Hua Lee. Watermarking in halftone images with parity-matched error diffusion. *Signal Processing*, 91(1):126–135, 2011.
- [80] Bipin Kumar Gupta, D Haranath, Shikha Saini, VN Singh, and V Shanker. Synthesis and characterization of ultra-fine  $\text{y}_2\text{o}_3$ :  $\text{Eu}^{3+}$  nanophosphors for luminescent security ink applications. *Nanotechnology*, 21(5):055607, 2010.
- [81] T Haist and H.J Tiziani. Optical detection of random features for high security applications. *Optics Communications*, 147(13):173 – 179, 1998.
- [82] Basel Halak, Mark Zwolinski, and M Syafiq Mispan. Overview of puf-based hardware security solutions for the internet of things. In *Circuits and Systems (MWSCAS), 2016 IEEE 59th International Midwest Symposium on*, pages 1–4. IEEE, 2016.
- [83] Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.
- [84] Ghaith Hammouri, Aykutlu Dana, and Berk Sunar. Cds have fingerprints too. In *CHES*, volume 9, pages 348–362. Springer, 2009.

- [85] Ghaith Hammouri, Erdinç Öztürk, and Berk Sunar. A tamper-proof and lightweight authentication scheme. *Pervasive and mobile computing*, 4(6):807–818, 2008.
- [86] Ghaith Hammouri and Berk Sunar. Puf-hb: A tamper-resilient hb based authentication protocol. In *Applied Cryptography and Network Security*, pages 346–365. Springer, 2008.
- [87] Feng Hao, Ross Anderson, and John Daugman. Combining crypto with biometrics effectively. *Computers, IEEE Transactions on*, 55(9):1081–1088, 2006.
- [88] Ryan Helinski, Dhruva Acharyya, and Jim Plusquellic. A physical unclonable function defined using power distribution system equivalent resistance variations. In *Proceedings of the 46th Annual Design Automation Conference*, pages 676–681. ACM, 2009.
- [89] Anthony TS Ho and Feng Shu. A print-and-scan resilient digital watermark for card authentication. In *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on*, volume 2, pages 1149–1152. IEEE, 2003.
- [90] Daniel E Holcomb, Wayne P Burleson, Kevin Fu, et al. Initial sram state as a fingerprint and source of true random numbers for rfid tags. In *Proceedings of the Conference on RFID Security*, volume 7, page 2, 2007.
- [91] Nicholas J Hopper and Manuel Blum. A secure human-computer authentication scheme. 2000.
- [92] Bill Horne, Lesley Matheson, Casey Sheehan, and Robert E Tarjan. Dynamic self-checking techniques for improved tamper resistance. In *ACM Workshop on Digital Rights Management*, pages 141–159. Springer, 2001.
- [93] Jyun-Yao Huang, I-Hui Li, and I-En Liao. A software licensing authorization scheme based on hardware component identifiers. In *Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on*, volume 3, pages 1673–1676. IEEE, 2014.
- [94] Tanya Ignatenko, Geert-Jan Schrijen, Boris Skoric, Pim Tuyls, and Frans Willems. Estimating the secrecy-rate of physical unclonable functions with

- the context-tree weighting method. In *Information Theory, 2006 IEEE International Symposium on*, pages 499–503. IEEE, 2006.
- [95] Ronald S Indeck and Marcel W Muller. Method and apparatus for fingerprinting magnetic media, November 15 1994. US Patent 5,365,586.
  - [96] Intel Security. McAfee labs threats report. <https://www.mcafee.com/uk/resources/reports/rp-quarterly-threats-mar-2016.pdf>. Accessed: 2017-10-25.
  - [97] Internet Engineering Task Force (IETF). The websocket protocol. Web Link available at <https://tools.ietf.org/html/rfc6455> – accessed in 2017, September 2017.
  - [98] ISO. Paper and board determination of opacity (paper backing) diffuse reflectance method. ISO 2471:2008, International Organization for Standardization, Geneva, Switzerland, 2008.
  - [99] Nav Jagpal, Eric Dingle, Jean-Philippe Gravel, Panayiotis Mavrommatis, Niels Provos, Moheeb Abu Rajab, and Kurt Thomas. Trends and lessons from three years fighting malicious extensions. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 579–593, 2015.
  - [100] Mariusz H Jakubowski, Chit Wei Nick Saw, and Ramarathnam Venkatesan. Tamper-tolerant software: Modeling and implementation. In *International Workshop on Security*, pages 125–139. Springer, 2009.
  - [101] Roger G Johnston. Tamper-indicating seals. *American Scientist*, 94(6):515, 2006.
  - [102] Ari Juels, David Molnar, and David Wagner. Security and privacy issues in e-passports. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 74–88. IEEE, 2005.
  - [103] Ari Juels and Martin Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM Conference on Computer and Communications Security, CCS ’99*, pages 28–36, New York, NY, USA, 1999. ACM.

- [104] Alexandros Kapravelos, Chris Grier, Neha Chachra, Christopher Kruegel, Giovanni Vigna, and Vern Paxson. Hulk: Eliciting malicious behavior in browser extensions. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 641–654, 2014.
- [105] Vineeth Kashyap and Ben Hardekopf. Security signature inference for javascript-based browser addons. In *Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization*, page 219. ACM, 2014.
- [106] Kaspersky Security Bulletin. Overall statistics for 2016. [https://kasperskycontenthub.com/securelist/files/2016/12/Kaspersky\\_Security\\_Bulletin\\_2016\\_Statistics\\_ENG.pdf](https://kasperskycontenthub.com/securelist/files/2016/12/Kaspersky_Security_Bulletin_2016_Statistics_ENG.pdf). Accessed: 2017-10-13.
- [107] Stefan Katzenbeisser, Ünal Kocabas, Vladimir Rožić, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. Pufs: Myth, fact or busted? a security evaluation of physically unclonable functions (pufs) cast in silicon. *Cryptographic Hardware and Embedded Systems—CHES 2012*, pages 283–301, 2012.
- [108] Muhammad Khan, Muhammad Akram, and Naveed Riaz. A comparative analysis of software protection schemes. *International Arab Journal of Information Technology (IAJIT)*, 12(3), 2015.
- [109] Darko Kirovski. Toward an automated verification of certificates of authenticity. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 160–169. ACM, 2004.
- [110] Darko Kirovski. Toward an automated verification of certificates of authenticity. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 160–169. ACM, 2004.
- [111] Darko Kirovski. Anti-counterfeiting: Mixing the physical and the digital world. In *Towards Hardware-Intrinsic Security*, pages 223–233. Springer, 2010.
- [112] Masato Kiuchi and Kazuharu Saito. Frequency analysis for security printing lines. In *Proc. SPIE*, volume 4677, page 111, 2002.
- [113] Aswin Raghav Krishna, Seetharam Narasimhan, Xinmu Wang, and Swarup Bhunia. Mecca: a robust low-overhead puf using embedded memory array.

In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 407–420. Springer, 2011.

- [114] Pawan Kumar, Jaya Dwivedi, and Bipin Kumar Gupta. Highly luminescent dual mode rare-earth nanorod assisted multi-stage excitable security ink for anti-counterfeiting applications. *Journal of Materials Chemistry C*, 2(48):10468–10475, 2014.
- [115] Sandeep S Kumar, Jorge Guajardo, Roel Maes, Geert-Jan Schrijen, and Pim Tuyls. The butterfly puf protecting ip on every fpga. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 67–70. IEEE, 2008.
- [116] Chih-Jen Lee, Sheng-De Wang, et al. Fingerprint feature extraction using Gabor filters. *Electronics Letters*, 35(4):288–290, 1999.
- [117] Jae W Lee, Daihyun Lim, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on*, pages 176–179. IEEE, 2004.
- [118] RA Lee. Micro-technology for anti-counterfeiting. *Microelectronic Engineering*, 53(1-4):513–516, 2000.
- [119] Ling Li. Technology designed to combat fakes in the global supply chain. *Business Horizons*, 56(2):167–177, 2013.
- [120] Daihyun Lim, Jae W Lee, Blaise Gassend, G Edward Suh, Marten Van Dijk, and Srinivas Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, 2005.
- [121] Ching-Yung Lin and Shih-Fu Chang. Robust image authentication method surviving jpeg lossy compression. In *Storage and Retrieval for Image and Video Databases (SPIE)*, volume 3312, pages 296–307, 1998.
- [122] Hod Lipson and Melba Kurman. *Fabricated: The new world of 3D printing*. John Wiley & Sons, 2013.

- [123] Chengjun Liu and Harry Wechsler. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *Image processing, IEEE Transactions on*, 11(4):467–476, 2002.
- [124] Lei Liu, Xinwen Zhang, Guanhua Yan, and Songqing Chen. Chrome extensions: Threat analysis and countermeasures. In *NDSS*, 2012.
- [125] Keith Lofstrom, W Robert Daasch, and Donald Taylor. Ic identification circuit using device mismatch. In *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, pages 372–373. IEEE, 2000.
- [126] Laura Luciani and Michela Casini. Multiple level technique for high resolution ovds fabrication. In *Proc. SPIE*, volume 3956, pages 298–304, 2000.
- [127] Roel Maes. *Physically unclonable functions: Constructions, properties and applications*. PhD thesis, Katholieke Universiteit Leuven, 2012.
- [128] Roel Maes. *Physically Unclonable Functions*. Springer, 2013.
- [129] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Intrinsic pufs from flip-flops on reconfigurable devices. In *3rd Benelux workshop on information and system security (WISSec 2008)*, volume 17, page 2008, 2008.
- [130] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for sram pufs. In *CHES*, volume 9, pages 332–347. Springer, 2009.
- [131] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. Pufky: A fully functional puf-based cryptographic key generator. *Cryptographic Hardware and Embedded Systems—CHES 2012*, pages 302–319, 2012.
- [132] Roel Maes and Ingrid Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*, pages 3–37. Springer, 2010.
- [133] MagnePrint. Magneprint. <http://www.magneprint.com/>. Accessed: 2017-08-24.
- [134] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. In *Embedded systems design with FPGAs*, pages 245–267. Springer, 2013.

- [135] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. A systematic method to evaluate and compare the performance of physical unclonable functions. In *Embedded systems design with FPGAs*, pages 245–267. Springer, 2013.
- [136] Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Techniques for design and implementation of secure reconfigurable pufs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2(1):5, 2009.
- [137] Mehrdad Majzoobi, Masoud Rostami, Farinaz Koushanfar, Dan S Wallach, and Srinivas Devadas. Slender puf protocol: A lightweight, robust, and secure authentication by substring matching. In *Security and Privacy Workshops (SPW), 2012 IEEE Symposium on*, pages 33–44. IEEE, 2012.
- [138] Said Marouf and Mohamed Shehab. Towards improving browser extension permission management and user awareness. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on*, pages 695–702. IEEE, 2012.
- [139] Jeevan M Meruga, William M Cross, P Stanley May, QuocAnh Luu, Grant A Crawford, and Jon J Kellar. Security printing of covert quick response codes using upconverting nanoparticle inks. *Nanotechnology*, 23(39):395201, 2012.
- [140] Eric Métois, Paul Yarin, Noah Salzman, and Joshua R Smith. Fiberfingerprint identification. In *Proc. 3rd Workshop on Automatic Identification*, pages 147–154, 2002.
- [141] Eric Métois, Paul Yarin, Noah Salzman, and Joshua R Smith. Fiberfingerprint identification. In *Proc. 3rd Workshop on Automatic Identification*, pages 147–154, 2002.
- [142] Leo A Meyerovich and Benjamin Livshits. Conscript: Specifying and enforcing fine-grained security policies for javascript in the browser. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 481–496. IEEE, 2010.
- [143] Mozilla. Statistics Dashboard. <https://addons.mozilla.org/en-US/statistics> – accessed in Feb. 2017, February 2017.
- [144] Nalperiron Inc. Henry roberts, ceo emeritus. <https://www.nalpeiron.com/the-team.html>. Accessed: 2017-10-16.

- [145] Gleb Naumovich and Nasir Memon. Preventing piracy, reverse engineering, and tampering. *Computer*, 36(7):64–71, 2003.
- [146] Newcastle University. Ethical review process. We Link available at [https://www.ncl.ac.uk/res/research/gov-ethics/ethics\\_procedures/ethical-review-process/index.htm](https://www.ncl.ac.uk/res/research/gov-ethics/ethics_procedures/ethical-review-process/index.htm) – accessed in 2018.
- [147] OECD/EUIPO. Trade in counterfeit and pirated goods.
- [148] Carl Olsmats. *The business mission of packaging: packaging as a strategic tool for business development towards the future*. Åbo Akad. University Press House, 2002.
- [149] Satoshi Ono, Takeru Maehara, and Kazunari Minami. Coevolutionary design of a watermark embedding scheme and an extraction algorithm for detecting replicated two-dimensional barcodes. *Applied Soft Computing*, 46:991–1007, 2016.
- [150] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, and Akshay Wadia. Universally composable secure computation with (malicious) physically uncloneable functions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 702–718. Springer, 2013.
- [151] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.
- [152] Oxford Dictionary. Definition of tamper. <http://www.oed.com/view/Entry/197420>. Accessed: 2017-10-16.
- [153] Oxford Dictionary. Definition of tamper evident. <http://www.oed.com/view/Entry/197420?redirectedFrom=tamper-evident#eid19243483>. Accessed: 2017-10-16.
- [154] Erdinc Ozturk, Ghaith Hammouri, and Berk Sunar. Physical unclonable function with tristate buffers. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 3194–3197. IEEE, 2008.
- [155] Erdinç Öztürk, Ghaith Hammouri, and Berk Sunar. Towards robust low cost authentication for pervasive devices. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 170–178. IEEE, 2008.

- [156] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.
- [157] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.
- [158] Ravikanth S. Pappu. *Physical one-way functions*. PhD thesis, Massachusetts Institute of Technology, March 2001.
- [159] Ravikanth Srinivasa Pappu. *Physical one-way functions*. PhD thesis, Massachusetts Institute of Technology, 2001.
- [160] Marcus Peinado, Yuqun Chen, Paul England, and John Manferdelli. Ngscb: A trusted open system. In *Australasian Conference on Information Security and Privacy*, pages 86–97. Springer, 2004.
- [161] Tuan Q Pham, Stuart W Perry, Peter A Fletcher, and RA Ashman. Paper fingerprinting using alpha-masked image matching. *Computer Vision, IET*, 5(4):232–243, 2011.
- [162] Politico. Eus passport fraud epidemic. <http://www.politico.eu/article/europes-fake-forged-stolen-passport-epidemic-visa-free-travel-rights/>. Accessed: 2017-10-13.
- [163] Reinhard Posch. Protecting devices by active coating. *Journal of Universal Computer Science*, 4(7):652–668, 1998.
- [164] J Pournelle. Zenith z-100, epson qx-10, software licensing, and the software piracy problem, 1983.
- [165] Pwc Global. Industry findings: Financial services. <https://www.pwc.com/gx/en/issues/information-security-survey/financial-services-industry.html>. Accessed: 2017-10-13.
- [166] Charles Reis, Adam Barth, and Carlos Pizano. Browser security: lessons from google chrome. *Queue*, 7(5):3, 2009.
- [167] Charles Reis, Steven D Gribble, Tadayoshi Kohno, and Nicholas C Weaver. Detecting in-flight page changes with web tripwires. In *NSDI*, volume 8, pages 31–44, 2008.

- [168] Ronitt Rubinfeld. Batch checking with applications to linear functions. *Information Processing Letters*, 42(2):77–80, 1992.
- [169] Ronitt Rubinfeld. Designing checkers for programs that run in parallel. *Algorithmica*, 15(4):287–301, 1996.
- [170] Ulrich Rührmair, Srinivas Devadas, and Farinaz Koushanfar. *Security Based on Physical Unclonability and Disorder*, pages 65–102. Springer New York, New York, NY, 2012.
- [171] Ulrich Rührmair, Jan Sölter, and Frank Sehnke. On the foundations of physical unclonable functions. *IACR Cryptology ePrint Archive*, 2009:277, 2009.
- [172] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. Puf-enhanced rfid security and privacy. In *Workshop on secure component and system identification (SECSI)*, 2010.
- [173] Anil Saini, Manoj Singh Gaur, and Vijay Laxmi. The darker side of firefox extension. In *Proceedings of the 6th International Conference on Security of Information and Networks*, pages 316–320. ACM, 2013.
- [174] Anil Saini, Manoj Singh Gaur, Vijay Laxmi, Tushar Singhal, and Mauro Conti. Privacy leakage attacks in browsers by colluding extensions. In *International Conference on Information Systems Security*, pages 257–276. Springer, 2014.
- [175] Wiwi Samsul, Henri P Uranus, and MD Birowosuto. Recognizing document’s originality by laser surface authentication. In *Advances in Computing, Control and Telecommunication Technologies (ACT), 2010 Second International Conference on*, pages 37–40. IEEE, 2010.
- [176] M GhanaatPisheh Sanaei, H Zamani, B Emami Abarghouei, and A Ghadiri Hakimi. Online modules: Novel model in serial-based method of software copy protection. *International Journal Computer Science and Telecommunication*, pages 1–9, 2013.
- [177] Ruchir Y Shah, Prajesh N Prajapati, and YK Agrawal. Anticounterfeit packaging technologies. *Journal of advanced pharmaceutical technology & research*, 1(4):368, 2010.

- [178] Ashlesh Sharma, Lakshminarayanan Subramanian, and Eric A Brewer. Paper-speckle: microscopic fingerprinting of paper. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 99–110. ACM, 2011.
- [179] Gustavus J Simmons. A system for verifying user identity and authorization at the point-of sale or access. *Cryptologia*, 8(1):1–21, 1984.
- [180] Gustavus J Simmons. Identification of data, devices, documents and individuals. In *Security Technology, 1991. Proceedings. 25th Annual 1991 IEEE International Carnahan Conference on*, pages 197–218. IEEE, 1991.
- [181] Boris Škorić, Stefan Maubach, Tom Kevenaar, and Pim Tuyls. Information-theoretic analysis of capacitive physical unclonable functions. *Journal of Applied physics*, 100(2):024902, 2006.
- [182] Boris Škoric, Pim Tuyls, and Wil Ophey. Robust key extraction from physical uncloneable functions. In *Applied Cryptography and Network Security*, volume 3531, pages 407–422. Springer, 2005.
- [183] Joshua R Smith and Andrew V Sutherland. Microstructure based indicia. In *Proceedings of the Second Workshop on Automatic Identification Advanced Technologies*, pages 79–83, 1999.
- [184] Joshua R Smith and Andrew V Sutherland. Microstructure based indicia. In *Proceedings of the Second Workshop on Automatic Identification Advanced Technologies*, pages 79–83, 1999.
- [185] SourceForge. Finding checksums on the sourceforge site. <https://sourceforge.net/p/forge/documentation/Verifying%20downloaded%20files/>. Accessed: 2017-10-16.
- [186] Ying Su, Jeremy Holleman, and Brian Otis. A 1.6 pj/bit 96% stable chip-id generating circuit using process variations. In *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, pages 406–611. IEEE, 2007.
- [187] G Edward Suh, Dwaine Clarke, Blaise Gassend, Marten Van Dijk, and Srinivas Devadas. Aegis: architecture for tamper-evident and tamper-resistant processing. In *Proceedings of the 17th annual international conference on Supercomputing*, pages 160–171. ACM, 2003.

- [188] G Edward Suh and Sriniwas Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual design automation conference*, pages 9–14. ACM, 2007.
- [189] Symantec. Internet security threat report. <https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf>. Accessed: 2017-10-25.
- [190] Gang Tan, Yuqun Chen, and Mariusz H Jakubowski. Delayed and controlled failures in tamper-resistant systems. In *In Proceedings of 8th Information Hiding Workshop*. Citeseer, 2006.
- [191] Michael Reed Teague. Image analysis via the general theory of moments\*. *JOSA*, 70(8):920–930, 1980.
- [192] Mike Ter Louw, Jin Soon Lim, and Venkat N Venkatakrishnan. Enhancing web browser security against malware extensions. *Journal in Computer Virology*, 4(3):179–195, 2008.
- [193] The Economist. De la rue rethinks its strategy. <https://www.economist.com/news/business/21720342-governments-are-striking-hard-bargains-what-they-pay-cash-de-la-rue>. Accessed: 2017-08-31.
- [194] The Wall Street Journal. Syrians seeking asylum in west use fake passports along the way. [https://www.wsj.com/article\\_email/syrians-seeking-asylum-in-west-use-fake-passports-along-the-way-1446420293-l](https://www.wsj.com/article_email/syrians-seeking-asylum-in-west-use-fake-passports-along-the-way-1446420293-l). Accessed: 2017-10-13.
- [195] Ken Thompson. Reflections on trusting trust. *Communications of the ACM*, 27(8):761–763, 1984.
- [196] Ehsan Toreini, Siamak F Shahandashti, and Feng Hao. Texture to the rescue: Practical paper fingerprinting based on texture patterns. *ACM Transactions on Privacy and Security (TOPS)*, 2017.
- [197] Hing-Chung Tsang, Moon-Chuen Lee, and Chi-Man Pun. A robust anti-tamper protection scheme. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 109–118. IEEE, 2011.

- [198] Pim Tuyls and Lejla Batina. Rfid-tags for anti-counterfeiting. In *Cryptographers Track at the RSA Conference*, pages 115–131. Springer, 2006.
- [199] Pim Tuyls, Tom Kevenaar, et al. *Security with noisy data: on private biometrics, secure key storage and anti-counterfeiting*. Springer Science & Business Media, 2007.
- [200] Pim Tuyls, Geert-Jan Schrijen, Boris Škorić, Jan Van Geloven, Nynke Verhaegh, and Rob Wolters. Read-proof hardware from protective coatings. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 369–383. Springer, 2006.
- [201] Pim Tuyls and Boris Skoric. Secret key generation from classical physics: Physical uncloneable functions. *AmIware: Hardware Technology Drivers of Ambient Intelligence(Philips Research Book Series*, 5:421–447, 2006.
- [202] Pim Tuyls and Boris Škorić. Strong authentication with physical unclonable functions. In *Security, Privacy, and Trust in Modern Data Management*, pages 133–148. Springer, 2007.
- [203] Pim Tuyls, Boris Skoric, Sjoerd Stallinga, Anton HM Akkermans, and Wil Ophey. Information-theoretic security analysis of physical uncloneable functions. In *Financial Cryptography*, volume 3570, pages 141–155. Springer, 2005.
- [204] Pirn Tuyls and Boris Skoric. Physical unclonable functions for enhanced security of tolcens and tags. *vieweg-it*, page 30, 2006.
- [205] Frerik van Beijnum, EG van Putten, KL van der Molen, and AP Mosk. Recognition of paper samples by correlation of their speckle patterns. *arXiv preprint physics/0610089*, 2006.
- [206] Marten E Van Dijk. System and method of reliable forward secret key sharing with physical random functions, January 26 2010. US Patent 7,653,197.
- [207] Leo Van Hove. Modelling banknote printing costs: of cohorts, generations, and note-years. *Economic Modelling*, 46:238–249, 2015.
- [208] Paul C Van Oorschot. Revisiting software protection. In *International Conference on Information Security*, pages 1–13. Springer, 2003.

- [209] R.L. van Renesse. 3das: a 3-dimensional-structure authentication system. In *Security and Detection, 1995., European Convention on*, pages 45–49, May 1995.
- [210] Rudolf L Van Renesse. *Optical document security*. Artech House Publishers, 1998.
- [211] Serge Vrijaldenhoven. Acoustical physical uncloneable functions. Master's thesis, Eindhoven University of Technology, Eindhoven, 2003.
- [212] W3C. W3c dom4. Web Link available at <https://www.w3.org/TR/dom/> – accessed in 2016, October 2016.
- [213] W3C. Html living standard. We Link available at <https://html.spec.whatwg.org/> – accessed in 2017, Mary 2017.
- [214] W3C Browser Extension Community Group. Browser extensions (report 23 july 2017). Web Link available at <https://browserext.github.io/browserext/> – accessed in 2017, September 2017.
- [215] W3c Fetch Living Standard. Opening handshake. Web Link available at <https://fetch.spec.whatwg.org/> – accessed in 2017, September 2017.
- [216] W3c HTML Living Standard. Web application apis. Web Link available at <https://html.spec.whatwg.org/multipage/webappapis.html> – accessed in 2017, September 2017.
- [217] W3c HTML Living Standard. Web sockets. Web Link available at <https://html.spec.whatwg.org/multipage/web-sockets.html> – accessed in 2017, September 2017.
- [218] Hsi-Chun Wang, Wen-Hsin Liu, Chia-Long Chang, and Yi-Hui Chen. Design of halftone-based ar markers under infrared detection. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 6, pages 97–100. IEEE, 2008.
- [219] Lei Wang, Ji Xiang, Jiwu Jing, and Lingchen Zhang. Towards fine-grained access control on browser extensions. In *International Conference on Information Security Practice and Experience*, pages 158–169. Springer, 2012.

- [220] Hal Wasserman and Manuel Blum. Software reliability via run-time result-checking. *Journal of the ACM (JACM)*, 44(6):826–849, 1997.
- [221] Dong Ryeol Whang, Youngmin You, Weon-Sik Chae, Jeongyun Heo, Sehoon Kim, and Soo Young Park. Solid-state phosphorescence-to-fluorescence switching in a cyclometalated ir (iii) complex containing an acid-labile chromophoric ancillary ligand: Implication for multimodal security printing. *Langmuir*, 28(44):15433–15437, 2012.
- [222] Stephen Wiesner. Conjugate coding. *ACM Sigact News*, 15(1):78–88, 1983.
- [223] Ping Wah Wong and Nasir Memon. Secret and public key image watermarking schemes for image authentication and ownership verification. *IEEE transactions on image processing*, 10(10):1593–1601, 2001.
- [224] Yuankun Xue, Ji Li, Shahin Nazarian, and Paul Bogdan. Fundamental challenges toward making the iot a reachable reality: A model-centric investigation. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 22(3):53, 2017.
- [225] Minerva M Yeung and Fred Mintzer. An invisible watermarking technique for image verification. In *Image Processing, 1997. Proceedings., International Conference on*, volume 2, pages 680–683. IEEE, 1997.