Università degli Studi di Padova
Department of Information Engineering

Digital Forensics Notes
Technical Part

A1. Different phases and actors involved in the acquisition and handling of digital evidences

The phases are:

1. **Identification**: search, recognition, and documentation of all the possible digital evidences. We must look for devices that may contain relevant data, a systematic and careful search is needed in order to identify all the devices (camouflaged devices). When identifying the evidences we should also note if their state may change over time (volatile memory), if so we should prioritize them in the collection/acquisition process. If the devices has a network interface, be it active or inactive, we have to identify the possible devices it's communicating with (or has communicated with). Peripherals are also another important source of relevant data, we should look out for camouflaged devices. There is also a virtual component during this phase which is represented by NAS or Cloud drives. Encrypted/Hidden partition are relevant too.

2. **Collection (seizure)**: acquisition of the physical device. After all the digital evidences have been identified, the DES and DEFR must decide whether to proceed with collection (physical) or acquisition of the device based on circumstances. Collection or Seizure is the phase in which devices are physically brought from the crime scene to the forensics lab. Devices may be in different states (off or on) which require different procedures, everything must be documented including packaging. Some non digital evidences may be collected too, like post-it with passwords, cables, power supplies. DEFR and DES will adopt different techniques based on time, cost, situation, and document the reasons behind their choices. NB1: physically (re)moving a device (to take it to the lab) is not suggested unless you know exactly how to do so (and how to preserve it). NB2: excluding devices from collection has to be justified.

3. **Acquisition**: copying the data from the devices. The acquisition process implies creating a copy of the data, and of course it all requires documentation. We want also to record slack space, deleted files and metadata hence copy/paste or printing are not suitable, but we need (would like) bit stream imaging. Hash functions must be used to determine that the produced copy image is identical to the original one. If the creation of a copy cannot be identical to the original data then the differences must be noted. If the verification process cannot be carried out i.e. we are imaging a live system or there are bad sectors or

there is a limited amount of time, that must be explained and documented. If the source is too big to copy then the DEFR might choose to only acquire important parts. All the forensic analysis must be performed on the copy. Every choice must be documented, explained, and possibly repeatable. Cloning creates a functional one to one copy of the hard drive while imaging creates an archive of a hard drive that can be used to make a one to one copy.

4. **Preservation**: storage and conservation of the integrity of the digital evidence. We must prevent digital devices from alteration and tampering. DEFR will have to demonstrate that the evidences have not been modified since the collection/acquisition, if changes occurred then they must be documented and justified. Privacy is not to be ignored in this cases, we must protect the privacy of the evidence owner by giving access to the evidence only to authorized personnel (and to a bare minimum). Everything must be labeled reporting who what when why and unique identifier of the devices (i.e. serial number). We must do everything not to turn off the device and minimize the risks concerning transportation.

The involved roles are:

- **Digital Evidence First Responder (DEFR)**: authorized and qualified operator which acts first on a digital crime scene. He must seizure the crime area, identify the person in charge for that area, profile all the other authorized people.
- **Digital Evidence Specialist (DES)**: performs DEFR task and must have the skill to handle several technical issues, as network analysis, RAM acquisition and operative system knowledge.
- **Incident Response Specialist (IRS)**: operates first after the fact (it can not be someone in charge of the investigation, e.g., network manager after an attack).
- **Forensics Laboratory Manager (FLM)**: scientist trained to perform lab tests related to crime investigation.

A2. Describe MD5 and SHA hash functions, providing details about their use in the acquisition of disk images and in the Chain-of-Custody

Hash functions are non invertible functions d that map a file f into its digest $d(f)$ (or $\underline{d}$) where f can be of variable-length while $d(f)$ is a fixed size string of symbols or bytes.

There are many hash functions but in the context of digital forensics (in any critical context actually) we want hash function to satisfy the following 3 properties:

1. **Resilience to pre-image**: given the digest $\underline{d}$ of an unknown file, it must be computationally unfeasible to to find a file f such that $d(f) = \underline{d}$.
2. **Resilience to second pre-image**: given the digest $\underline{d} = d(f_1)$ and f_1, it must be computationally unfeasible to find a file f_2 such that $d(f_2) = d(f_1) = \underline{d}$.
3. **Resilience to collisions**: it must be computationally unfeasible to find f_1 and f_2 such that $d(f_2) = d(f_1)$.

MD5 and SHA1 are hash functions. In DF hash function are used to verify that two files, partitions, etc are completely identical. MD5 generates strings of length equal to 128 bits while SHA1 generates strings of length equal to 160 bits, unfortunately for both have been found ways to break one or more of the properties mentioned above.

This does not however make them useless, one could in fact use both of them to verify the integrity of a copy, the idea is that if the attacker managed to create a fake copy of a file with the same MD5 hash there are virtually no chances that the SHA1 hash of the two files do match too.

Hash function have of course to be used when creating forensic copies of disks or files of any kind: we want to be sure that whatever we are copying is identical to the source on a bit level. It is important of course to always verify the hash of a copy along the whole chain of custody to guarantee that no one altered the files.

A3. Overview the different steps of seizure and acquisition for a mobile device.

First of all we should isolate the device and prevent anyone from accessing it, we should take pictures just to have a reference of the phone's condition at collection time, physical defects must be noted too.

We should then collect the phones accessories like charging cables, earbuds or whatever we think was somehow linked to that device (instructions and box too, sim envelope for PIN and PUK). Of course everything should be documented, we should note who, when, where and why for each step.

If the device is switched off then we should leave it that way, but if it is on we must try to keep it on (using charging cable), since switching it off may cause loss of data and may enable some locking mechanism on the next startup. In case it is switched on we should also report what can be seen on the screen (time and date and offset wrt real time if any) but we should not interact with it (the least possible).

The device should also be isolated from any kind of network (i.e. WiFi, 3g, and mobile network) to prevent remote users from accessing it or from modifying its state. We can do so by enabling airplane mode (which unfortunately requires active interaction with the phone and knowledge about the specific model / OS) or by putting it in a Faraday cage which will drain the device's battery since it will try to connect continuously.

Mobile devices can store data in multiple places, SD Card, SIM card, Internal Memories. For SD cards we can apply the same techniques used for disk imaging, this means that we should try to create exact copies of the card without changing a single bit (verifiable via checksums).

For sim cards acquisition it is possible to use specialized tools and software like SIMSpy2, for integrated memory we can try to extract physically the chip or exploit the JTAG interface. Logic acquisition of the integrated memory can be performed using Samsung Kies or iTunes or custom hardware like cellbrite UFED.

A4. Eavesdropping on cabled and wireless networks: provide a description of the main sniffing strategies and their countermeasures.

On shared ethernet one can intercept packets simply by not discarding packets not directed to him, this can be done by enabling promiscuous mode on the network interface. On switched

ethernet this approach cannot work since the packets are correctly routed by the switch, we can bypass this form of protection by exploiting the ARP protocol which being based on UDP is connectionless (it does not require a handshake between the two endpoints). This means that an attacker could perform an attack called ARP spoofing in which he basically "tells everyone" (without being asked by anyone) that he is the owner of an arbitrary MAC address: in this way he will be able to intercept packets sent to that MAC address since those packets will be routed to him.

The attacker can also exploit the limited amount of storage used for the ARP cache, he can do so by flooding the switch with a huge amount of ARP requests; the switch will not be able to keep it up with the requests and will therefore activate failopen mode, in which the network becomes identical to a scenario with shared ethernet. These techniques can be replicated easily in wireless networks too. Some of the tools that can be used for such activities are `tcpdump`, `wireshark`, `fiddler`, `snort`, `ettercap`.

A sniffer can be detected using multiple methods:

- **ping method**: send a packet to the suspected machine including in the packet a wrong MAC address. If you receive a response then you found the sniffer: no legit machine would answer to you if you send a packet to them with an incorrect MAC address.
- **ARP method**: send a non broadcast ARP response (which will be captured by anyone in promiscuous mode i.e. the sniffer). Then send a broadcast ping request to our own IP but with a different MAC: only the machine that previously sniffed our ARP request will be able to respond. If it does so, then you found the sniffer.
- **Latency method**: sniffers will parse every packet; flood the network with data (possibly with useless packets, not directed to anyone), the sniffer will parse them; at the same time start pinging the suspected sniffer, if its response time worsens during the flooding then it's because it is probably parsing all the packets so you found the sniffer. (This method could return many false positives).
- **Others**: check your interfaces for hidden sniffers via `ifconfig`, use `arpwatch` to monitor ARP cache, use IDS to monitor IP/MAC pairs and alert when finding a mismatch.

We can also use crypto (ssl/tls, ssh, gnupg) to prevent sniffers from reading our data/mail; we could also enforce more strict rules on MAC addresses, etc.

A5. Present the Dempster-Shafer theory and explain how it can be employed in sensor fusion.

Dempster-Shafer theory is a general framework for reasoning with uncertainty.

Consider a scenario where each sensor has a degree of support for the fact: 0 means no support and 1 means full support.

Let X be the universe (set of possible states/conclusions for the analyzed system).

An event $A \in X$ is **mutually exclusive** (at most, one is true) and **exhaustive** (at least, one is true).

The power set Θ is the set of all possible subsets of X including the empty set. It must always hold:

- $\Pr[\emptyset] = 0$ (at least one of the options must be true)
- $0 \leq \Pr[A] \leq 1$

- $\Pr[X] = 1$ (only one outcome must be true)

For example, consider $X = \{\text{attack, normal}\}$.
Then $\Theta = \{\emptyset, \{\text{attack}\}, \{\text{normal}\}, \{\text{normal, attack}\}\}$.
The last subset {normal, attack} indicates uncertainty.
The Basic Belief Assignment (BBA) is a mass function $m(A) : 2^x \to [0, 1]$.
It holds:

- $m(\emptyset) = 0$
- $\sum_{A \in 2^X} m(A) = 1$

The mass $m(A)$ of A expresses the proportion of all relevant and available evidence that supports the claim that the actual state belongs to A but to no particular subset of A (they have their own mass).

For example, $m(A \vee B) = 0.3$ means that there is evidence for $\{A \vee B\}$ that can not be divided into evidences for A or B.

The belief (support) function is $bel(A) = \sum_{B:B \subseteq A} m(B)$.

The plausibility function is $pl(A) = \sum_{B:B \cap A \neq \emptyset} m(B)$.

Given two source of evidence, it is possible to combine them in order to generate a belief value:

$$m(C) = \sum_{A_i \cap B_j = C} m_1(A_i) m_2(B_j) \tag{1}$$

The normalization factor is

$$K = \sum_{A_i \cap B_j \neq \emptyset} m_1(A_i) m_2(B_j) \tag{2}$$

A6.

A7.

A8. Camera ballistic: describe how PRNU can be extracted and used to identify a specific camera.

When hit by uniform light we expect each camera sensor to output exactly the same value, this is not true in practice, because of some small variations in size and material we get slightly different values. This difference however is consistent, this means that each camera introduces some kind of fixed noise into the photos which can be seen as a unique signature for that camera. PRNU (Photo Response Non Uniformity) is the difference between the ideal response of the sensors and the true response.

An image can be represented as $\mathbf{I} = \mathbf{I}^{(0)} + \mathbf{I}^{(0)}\mathbf{K} + \mathbf{\Theta}$, where $\mathbf{I}^{(0)}$ is the ideal sensor output (without noise), $\mathbf{K}$ is the PRNU fingerprint of the camera and $\mathbf{\Theta}$ accounts for all the other types of noise.

Pattern noise can be estimated by taking the difference between an image $\mathbf{I}$ and its denoised version: $\mathbf{W_I} = \mathbf{I} - F\left(\mathbf{I}^{(0)}\right)$, where $\mathbf{W_I}$ is called *residual noise* and F is a *denoising filter*.

The PRNU fingerprint can be estimated using many images and a maximum likelihood estimator: $\hat{\mathbf{K}} = \frac{\sum_{i=1}^{N} \mathbf{W}_i \mathbf{I}_i}{\sum_{i=1}^{N} (\mathbf{I}_i)^2}$.

If we are given a new image $\mathbf{J}$ we can determine the presence of $\mathbf{K}$ by computing the correlation detector: $\rho = corr\left(\mathbf{W_J}, \hat{\mathbf{K}}\mathbf{J}\right)$

A9. Overview of some possible image tampering detection strategies

Every device leaves a mark when acquiring a photo. Softwares (like Photoshop) and compression algorithms do it as well. If we can detect such marks we can detect if an image has been tampered with. If we find different PRNUs or different JPEG compression footprints (blocking artifacts, DCT coefficients statistics, double compression) in the same image we could infer that it has been modified.

We can detect alterations in an image using **Benford's law**, which states that the distribution of the first digit of a given set of numeric data should follow a particular shape. Benford says that the first digit is "likely" to be small. This usually works in practice during experiments but is hard to demonstrate formally. I.e. if we notice that an image does not follow Benford's law we cannot instantly assume that it is forged, but it may still be a good indicator that further investigation is needed.

Another way to detect tampering on an image is to look for **JPEG artifacts**, if we see an area in which these artifacts are different from the rest of the image we have a good indicator that something has been done in that area, maybe a cut and paste plus double compression. We can detect cut and paste duplicates by scanning the image with small windows: if we find that two windows are very similar we are probably encountering a duplicate element.

SIFT (Scale-Invariant Feature Transform) works more or less in the same way: it finds key-points of objects extracted from sample images and saves them in a database. When we encounter a new image we scan the image in search of similar objects we have saved in the database and compare the distances between the features using the euclidean distance.

A10. Describe how additive spread-spectrum watermarking works and its possible application.

Spread spectrum watermarking works by inserting noise-like information in images. The coordinates in which the "noise" is added are not random. Since we want to preserve the correct perception of the image (i.e. we want the watermark not be visible) we add low amounts of noise which can be increased if we exploit the Human contrast sensitivity which states that the human perception system is less sensitive to extremely gradual changes and to extremely fast changes. Edges are usually good for hiding information (they also tend to be preserved when using compression which allows the watermark to be resilient to this kind of attacks).

We can usually perform some kind of perceptual analysis of the image to better understand better where we can hide information. To perform the watermarking we need a key, this key will allow us to divide the image in specific disjoint tiles. We then combine the perceptual mask value of a pixel with a number extracted randomly from a Gaussian distribution (mean=0, var=1); the combination is then linearly combined with a bit of the message we are trying to

hide (this is w, the watermark). This is done for each pixel in the tiles. The final image is composed by the sum of pixel value of the original image plus the watermark we computed for that pixel.

To detect the presence of the watermark and decode the hidden message one would need to know the key and the pseudo random sequence used (number sequence extracted from the Gaussian).
Applications of spread spectrum can be video/audio/software/text/executable watermarking, labeling, fingerprinting authentication, copy and playback control.
Attacks can be anything: additive noise, filtering, cropping, compression, rotation and scaling, multiple watermarking.

A11. Generative Adversarial Networks (GAN): definition, cost function to be minimized and applications.

The GAN framework consists in a minimax game played by two neural networks. The first network is a generative model which takes as input random noise and the second network is a discriminative model which aims to distinguish real samples from samples originated by the generator. The generator instead aims to produce samples indistinguishable from the real ones and therefore to "fool" the discriminator.

Let p_g be the generator's distribution and $p_z(\mathbf{z})$ be the noise distribution. The generator learns a mapping $G(\mathbf{z})$ from the noise to the data space, so its output distribution p_g is desired to be similar to the one of the data, p_{data}.

The discriminator learns the function $D(\mathbf{x})$, which represents the probability that $\mathbf{x}$ comes from the data distribution.

The generator is trained to minimize the loss

$$L_G = \log\left(D(1 - G(\mathbf{z}))\right) \tag{3}$$

while the discriminator is trained to maximize the loss

$$L_D = \log\left(D(\mathbf{x})\right) + \log\left(1 - D(G(\mathbf{z}))\right) \tag{4}$$

The mini-max game is given by

$$\min_G \max_D E_{\mathbf{x}\sim p_{data}}\left[\log\left(D(\mathbf{x})\right)\right] + E_{\mathbf{z}\sim p_z}\left[\log\left(1 - D(G(\mathbf{z}))\right)\right] \tag{5}$$

The advantages of GANs are:

- few restrictions on the generator;
- no need for Markov model;
- no overfitting, since input is random noise.

The disadvantages of GANs are:

- slow learning and training, since input is noise;
- discriminator may outperform the generator, therefore there is no improvement between subsequent iterations.