**Guideline & Installation**

**Module A.2 - FPGA based Visual Capture Device**

DeepShield: A distributed and cryptographic approach for authenticating digital content, based on Trusted Execution environments (TEE), establishing ownership, and detecting unwanted manipulations with focus on the Field Programmable Gate Array (FPGA) based visual capture device with integration with the overall DeepShield approach.

Stage 1 (Months 1–7, timeboxed, final demonstration 28th Mail 2025):

Guideline how to setup and use the FPGA based visual capture device (Module A.2) demonstrated for phase Stage 1.

|  |  |
| --- | --- |
| Document No.: | **A2\_FPGA\_VCD\_Guideline** |
| Issue: | A |
| Date: | 05.05.2025 |

|  |  |
| --- | --- |
|  |  |

**Revision Record**

|  |  |  |  |
| --- | --- | --- | --- |
| **Issue** | **Date** | **Description / Change** | **Modified Chapters** |
| A | 12.05.2025 | Delivered version | All |

Contact: langewisch@secublox.com

# **Table of contents**

[Table of contents 2](#_Toc197250221)

[Introduction 3](#_Toc197250222)

[Overall Goal DeepShield Approach 3](#_Toc197250223)

[Focus Module A.2 3](#_Toc197250224)

[List of Abbreviations 4](#_Toc197250225)

[DeepShield Overall Approach 5](#_Toc197250226)

[Important points upfront 6](#_Toc197250227)

[Software Setup 7](#_Toc197250228)

[Install Apple Command Line Tools 7](#_Toc197250229)

[Install “brew” 7](#_Toc197250230)

[Install “brew packages” 7](#_Toc197250231)

[Build FPGA based Visual Capture Device Software 8](#_Toc197250232)

[Setup – FPGA based Visual Capture Device 9](#_Toc197250233)

[Install Lens 9](#_Toc197250234)

[Connect Ethernet 9](#_Toc197250235)

[Install Power Supply 9](#_Toc197250236)

[Connect Camera to Mac 10](#_Toc197250237)

[Run – FPGA based Visual Capture Device 11](#_Toc197250238)

[First Time Setup 11](#_Toc197250239)

[Demonstrate – FPGA based Visual Capture Device 12](#_Toc197250240)

[Find the camera 13](#_Toc197250241)

[Basic Adjustments 14](#_Toc197250242)

[Streaming 15](#_Toc197250243)

[Demonstrate – FPGA based Visual Capture Device: Capture a frame 16](#_Toc197250244)

[Develop your own App 16](#_Toc197250245)

# Introduction

The rapid development of digital technologies in recent years has opened both fascinating and unsettling possibilities, particularly **in** synthetic content and deepfakes. This sophisticated AI-generated audio, image and video content is now so realistic that it is almost indistinguishable from real footage.

With the exponential increase in the performance of systems and language models based on artificial intelligence (AI) and machine learning (ML), not only the quantity but the quality of deepfakes is improving rapidly. This poses dangers – for trust, security and the perception of reality in our society.

Efforts to combat deepfakes are currently focused on two major areas:

* **Detection:** Development of AI algorithms to identify deepfakes
* **Prevention:** Implementation of authentication mechanisms for digital content.

Despite significant progress in these areas, major challenges remain, such as the generalizability and scalability of AI deepfake detection systems and the establishment of a manipulation-resistant standard for image metadata.

## Overall Goal DeepShield Approach

Development of a prototype that reliably detects deepfakes images from various media content and/or protects existing infrastructures from the use of deepfakes with preventive measures. To address these goals the overall DeepShield approach was divided between different independent modules (A.1, A.2, B, C, D) and will be integrated to complete the overall DeepShield approach, demonstrated at the end of stage 1.

## Focus Module A.2

This document focusses on the guideline to setup and use module A.2 a prototype demonstration of the FPGA based visual capture devices integrated into the overall DeepShield approach.

|  |  |
| --- | --- |
| Several black cubes with different components  AI-generated content may be incorrect. |  |

Figure 1: DeepShield – Stage 1 – A.2 – FPGA Device and Application

This document focuses on the setup and use of module A.2.

## List of Abbreviations

Below in Table 1 is a list of abbreviations used in this document.

|  |  |
| --- | --- |
| **Abbreviation** | **Description** |
| BC | Blockchain |
| DLT | Distributed Ledger Technology |
| DN | Decentralized Network |
| DSC | DeepShield |
| IPFS | InterPlanetary File System |
| KYC | Know Your Customer |
| LLM | Large Language Model |
| NFT | Non-Fungible Token |
| PRNG | Pseudo-Random Number |
| TEE | Trusted Execution Environments |
| TRL | Technology Readiness Level |
|  |  |
|  |  |
|  |  |

## DeepShield Overall Approach

The DeepShield approach will focus on following requirements to demonstrate the defined goals by following functionalities:

|  |  |
| --- | --- |
| **A: Content Creation & Secure Watermarking** |  |
| A user captures a photo or video with on a public blockchain registered device.  These photos or videos are watermarked on-device within a secure, trusted execution environment making use of the private device key and then stored locally or elsewhere. |  |
|  |  |
| **B: Content Registration**  Additionally, to the preregistered public key of the devices, the original watermarked content can be registered on a public blockchain. This allows for ensuring context integrity using further meta information such as reference vector embeddings. |  |
| **C: Content Verification & Context Integrity**  The registered public key enables verification of the content’s authenticity, confirming its origin from a specific device based on the watermark. Degradations in the watermark indicate content manipulation.  Using meta information such as reference vector embeddings, contextual integrity can be verified, in a multilayered approach, helping to prevent disinformation. |  |
| **D: Verification and Revocation via Distributed Ledger**  Public Key Management: Verifiers can retrieve the device’s public key from the blockchain to authenticate the watermark and digital signature, ensuring transparent, verifiable content authenticity.  Revocation Notices: The ledger will contain information about revoked or compromised keys, ensuring that only active and trusted devices can be verified  Smart Contracts for Automation: Implement smart contracts to handle device registration, key updates, and revocation seamlessly |  |

## Important points upfront

1. **Network:** the provided Secublox cameras are in the network 192.168.1 - as many standard networks, like Wifi at home or office. We can change that later, no issue, but for this setup we are stuck with that. Therefore you need to make sure EITHER a) your wifi (or any other network interface using 192.168.1) is OFF - or - b) you assign a network other than 192.168.1. to the Wifi or other network you have.
2. **MacOS** is very protective - and to get around that quickly, is to compile the software we provide on all the Macs you are going to use, individually. DO NOT copy the binary you compiled on one Mac to the other - it will not work. Further, on each Mac you have to click through a few Security Settings, maybe even 2 times - not a big issue, just needs to be done once. Instructions below.
3. **Sudo Rights:** make sure the Mac user has “sudo” rights. Try in a Terminal/shell “sudo ls” to see if it works. If not – follow instructions given by Google, eg. here:

<https://apple.stackexchange.com/questions/406429/why-can-i-not-use-sudo-as-a-standard-user-in-macos>

# Software Setup

## Install Apple Command Line Tools

1. In a MacOS “terminal” or “shell window” execute the following command:

xcode-select –install

1. or install “Xcode” from the App Store and follow all instructions.

## Install “brew”

1. Follow instructions here: <https://brew.sh>

## Install “brew packages”

1. In a MacOS Terminal/shell execute the following command:

brew install cmake jpeg-turbo wolfssl wxwidgets

If your Terminal/shell cannot find “brew” – try this:

/opt/homebrew/bin/brew install cmake jpeg-turbo wolfssl wxwidgets

## Build FPGA based Visual Capture Device Software

1. Safe the ZIP file in a folder of your choice.
2. Unzip the file (double click it). It creates a new folder, in which all the relevant files are stored.
3. Open a Terminal/shell in this new folder.
4. Execute the following commands in the that Terminal/shell - these steps build all necessary files for the creation of the executable.

mkdir build

cd build

cmake -DCMAKE\_BUILD\_TYPE=Release ..

If your Terminal/shell cannot find “cmake” – try this:

/opt/homebrew/bin/cmake -DCMAKE\_BUILD\_TYPE=Release ..

1. Now we create the executable:

make

1. After this step the executable is available here:

ls -al SecubloxDemo/SecubloxDemo

Will show you a file named SecubloxDemo – this is the executable we will use later.

# Setup – FPGA based Visual Capture Device

## Install Lens

1. Make sure the aperture of the lens is fully open. Look through the lens, rotate the according handle, and use the setting with the most wide-open setting.
2. Remove the rubber camera sensor protection from the camera housing. Store the rubber somewhere safe, to use it when the camera is without the lens, to protect the camera sensor.
3. Carefully mount the lens on the camera housing – screw it not too tight, but also not too loose.

## Connect Ethernet

1. As simple as with any other Ethernet cable – plug it in to the Secublox Camera.

## Install Power Supply

**IMPORTANT:** ALWAYS connect the power supply TO THE CAMERA first, before you plug in the power adapter into the mains socket or switch the camera power adaptor on.

1. Just connect the connector to the cameras power input, making sure the camera power adaptor is switched off and/or not connected to the mains power supply.
2. Plug the camera power adaptor into mains power supply and switch the camera power adaptor on.

**IMPORTANT:** When disconnecting FIRST switch the camera power adaptor off or pull it from the mains socket and then disconnect the power supply from the camera.

## Connect Camera to Mac

1. Connect the other end of the Ethernet cable to the Ethernet port on the Mac side (your USB-C Ethernet adapter).
2. On the Mac make sure the network for this interface is 192.168.1 and the Mac has an IP address assigned on this interface – e.g. 192.168.1.200
3. You can check/list the interfaces like this in a Terminal/shell:

ifconfig

1. Find the interface which is related to the USB-C adapter – this may vary from each Mac to Mac, usually it is enX – X any number. Set the IP address for interface en6 as example:

sudo ifconfig en6 192.168.1.200

**IMPORTANT:** This is a temporary assignment of the IP address to this interface and has to be repeated before any setup (eg. when the Mac is restarted, or the USB-C Ethernet adapter has been disconnected from the Mac). If you want a permanent assignment – check with us, though we don’t recommend at this stage.

# Run – FPGA based Visual Capture Device

## First Time Setup

There are a few steps necessary ONLY 1st time the Secublox Camera Software is run on a Mac due to security restrictions on MacOS and due to the current experimental setup.

1. Open “Settings” and go to “Privacy & Security”
2. Scroll to the very end – this is very important to do that!
3. When run the 1st time, we will now get all security approvals from MacOS accomplished. Several warning windows in sequence will pop-up and you need to take action in the “Setting” app. After the creation of the executable (see above Build Secublox Camera Software), you still have the Terminal/shell open. Switch to that Terminal/shell and run the following command:

./SecubloxDemo/SecubloxDemo

1. In the “Setting” App you will see a new item created, and there you need to click on “Allow Anyway” and the “Done” (DO NOT click “Move to Trash”) in the pop-up window. This will happen a few times for different libraries we have linked.

An example is here:

A screenshot of a phone

AI-generated content may be incorrect.

Figure 1: DeepShield – Stage 1 – A.2 – Privacy & Security

1. In case the SecubloxDemo App crashes or closes, start it again:

./SecubloxDemo/SecubloxDemo

# Demonstrate – FPGA based Visual Capture Device

1. Now that all MacOS security approvals have been done – if needed start the Secublox

Camera Software again:

./SecubloxDemo/SecubloxDemo

This will open several windows for controlling, streaming data, log information and frame information. Arrange them in a way like shown here:

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1: DeepShield – Stage 1 – A.2 – FPGA Device Application

## Find the camera

1. Switch to the Tab “Device” in the “Device Properties” window (right window in the above example) and click on “Get Device List”. The result should be an IP Address and a filled-out form as in the example below. That means the camera has been found.
2. If the camera has not been found, make sure the network settings are correct, and the camera is on. Follow the steps discussed before.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1: DeepShield – Stage 1 – A.2 – FPGA Device Application - Device

## Basic Adjustments

1. Go to the Tab “Signal” and set the camera to approximately the settings shown next. Doesn’t need to be precise, it just makes sure the stream/frame is in a good range. Adjust that later to the “best” setting for the existing environment.
2. Go to the Tab “Compress” and set the “Compression Level” to “2”. For this application, this is the best setting.

Screens screenshot of a computer

AI-generated content may be incorrect.

Figure 1: DeepShield – Stage 1 – A.2 – FPGA Device Application - Compress

## Streaming

1. Click the marked button – this is to start and stop the video stream. In the “Cam Log Window” you will also see “Start stream” or “Stop stream” respectively.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1: DeepShield – Stage 1 – A.2 – FPGA Device Application - Streaming

1. Adjust the rulers of the “SecuBlox Viewer” window to the “middle postion” to have the window centered in the middle of the camera frame or increase the size of the window.
2. Click the button marked, right of “Start/stop Stream”.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1: DeepShield – Stage 1 – A.2 – FPGA Device Application – Streaming Start/Stop

1. You should see the video output in the window. If not, close the App and start over.

# Demonstrate – FPGA based Visual Capture Device: Capture a frame

1. Click on “Capture” and just follow the dialog to safe the captured frame in the directory of your choice. The frame will be stored in raw format (frame and metadata) and encrypted (frame and metadata).

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1: DeepShield – Stage 1 – A.2 – FPGA Device Application – Capture Image

# Develop your own App

We provided the necessary “brew instructions” and libraries together with source code, which demonstrates how to use the API/libraries for all the functions necessary for the April demo, as well as how to build and integrate an executable from source code with “cmake” configuration files.

All can be easily adjusted, re-used for own specific app/software development.