Signed File Transfer

mit Kunbus Revolution Pi
&
ATECC508A Security Chip

03.02.2019

Gruppe 1

Peter Lang, Harald Bogesch, Felix Meißner, Christian Remmele

Inhaltsverzeichnis

1 Einleitung

IoT

Durch die zunehmende Digitalisierung ist man in der heutigen Zeit in der Lage, verschieden Geräte im eigenen Heim zu vernetzen und daraus diverse Vorteile zu ziehen. Zum Beispiel ist es ganz einfach, mit dem Smartphone von unterwegs die Heizung zuhause einzuschalten. Nicht nur im Privaten Leben ist dies möglich, sondern auch in der Industrie kommunizieren verschiedene Geräte, Systeme und sogar komplexe Fertigungsanlagen über Netzwerke wie das Internet miteinander. Immer wieder taucht in diesem Zusammenhang der Begriff Internet of Things (dt. Internet der Dinge) auf. Doch was ist unter dem Begriff Internet of Things (im folgenden IoT genannt) zu verstehen?

Mit dem IoT ist es möglich, industrielle Systeme über Netzwerke, wie das Internet, miteinander kommunizieren zu lassen, ohne dass dafür eine Person benötigt wird. Dadurch können Daten ausgetauscht werden, Systeme überwacht werden oder Aktionen automatisch angestoßen werden. Als Kommunikationspartner dienen oft eingebetteten Systeme (engl. Embedded Systems). Eingebettete Systeme sind Maschinen oder Geräte, in deren technischen Kontext ein elektronischer Rechner oder PC eingebunden ist. Bei der direkten Kommunikation zwischen Geräten spricht man auch oft von der M2M-Kommunikation (Maschine zu Maschine).

Dieser maschinellen Kommunikation und den dabei auftretenden Herausforderungen widmet sich die folgende Arbeit.

Die Struktur dieses Dokuments ist wie folgt aufgebaut:

- **Motivation und Problemstellung**
  - Auftretende Probleme der offenen Netzwerkkommunikation
  - Grundaufbau und Szenario

- **Prinzipdarstellung**
  - Signatur und Verifikation
  - Schutzziele und Sicherheitsniveau

- **Aufbau und Vorbereitung der Arbeit**
  - Hardwareaufbau
  - Notwendige Softwarepakete

- **Funktionsumfang des benutzten ATECC508A und Grundkonfiguration**

- **Programmbeschreibung des „Signed File Transfer“**
  - Sender
  - Empfänger

- **Abschließendes Fazit**

1.1 Motivation und Problemstellung

IoT hat viele Vorteile, jedoch müssen seine Anwendungsfälle auch so gut wie möglich gegenüber Angreifern geschützt sein. Dadurch, dass die meisten Geräte mit dem Internet verbunden sind ist es nämlich durchaus möglich, dass von außerhalb Nachrichten manipuliert und Daten verändern werden können. Das Grundszenario dieser Arbeit (Abbildung 1) ist demnach wie folgt aufgebaut:

- **Produktionsanlage**
  - stellt z.B. ein Bauteil her
  - Bauteil wird am Ende des Produktionsprozesses für die Qualitätssicherung fotografiert.
  - Die entstandene Bilddatei wird zusammen mit ein paar zugehörigen Sensorwerten an eine Überwachungsstation gesendet.

- **Anlagenüberwachung**
  - Die Überwachungsstation kann aufgrund der Netzwerkanbindung der Produktionsanlage an einem beliebigen physikalischen Ort platziert werden
  - Daten werden von der Produktionsanlage empfangen und nach Bedarf bearbeitet oder archiviert

- **Angreifer**
  - Hat physikalischen Zugang zum Netzwerk
  - Möchte Daten böswillig verändern und der Produktion zu schaden
  - Möchte die Dokumentation von schlechten Prozessergebnissen verhindern
  - ….

- **Kommunikation**
  - Über eine undefinierte Netzwerkstruktur

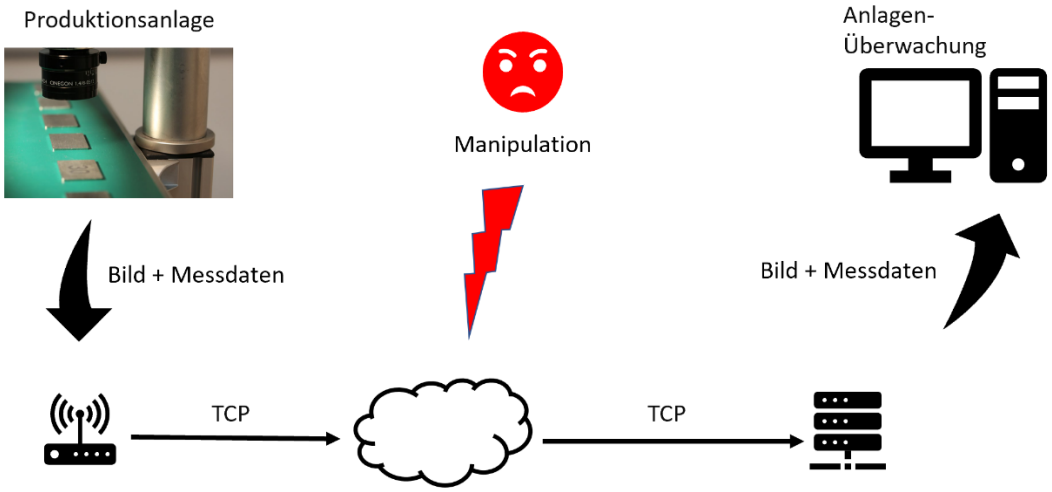

Abbildung 1: Grundaufbau

Die Kommunikation der Teilnehmer kann sehr einfach manipuliert werden, da die gängigen Netzwerkstandards wie z.B. TCP-IP (Transportprotokolle des Internets) nicht genügend Schutz dagegen bieten. Somit wäre es der Anlagenüberwachung nicht möglich, sicherzustellen ob die Daten tatsächlich von der Produktionsanlage stammen oder ob sie unterwegs manipuliert wurden. Die Daten wären deshalb im Zweifelsfalle nicht belastbar.

Im Konkreten sind folgende Problemstellungen zu betrachten:

- **Daten der Produktionsanlagen müssen gegen Manipulationen geschützt werden**
  - wie können Daten geschützt werden?
  - wie kann das geprüft werden?
  - stammen die Daten tatsächlich von der Produktionsanlage?

- **Kommunikationskanal muss gesichert werden**
  - wie kann die Verbindung gesichert werden?
  - wird zusätzliche Hardware und Software benötigt?

- **Sicherheitsniveau soll dem aktuellen Stand der Technik entsprechen**

Ziel ist es, die simulierten Daten von der Produktionsanlage zu signieren und an die Anlagenüberwachung zu schicken. Die Anlagenüberwachung soll anschließend die Signatur überprüfen und eine Plausibilitätsprüfung der zusätzlichen Merkmale durchführen. Die Signatur fungiert hierbei wie eine Unterschrift auf einem handgeschriebenen Brief. Mit einer Unterschrift stellt man sicher, dass die auf dem Dokument befindlichen Informationen wirklich vom Unterzeichner stammen und dieser auch mit diesen einverstanden ist. Wie bei einem Brief gibt es auch bei den Digitalen Nachrichten Informationen die für eine Nachvollziehbarkeit erforderlich sind. Beispiele hierfür sind Datum und Uhrzeit oder einfach der Name des Senders.

Um dies digitale Unterschrift mit Briefkopf zu verwirklichen, werden zwei Revolution Pi Core 3 der Firma Kunbus verwendet (Abbildung 2). Diese fungieren als Kommunikationsschnittstelle in der Produktionsanlage und der Anlagenüberwachung. Durch den ATECC508A Security Chip, der in dem Revolution Pi integriert ist, sollen die Nachrichten signiert und verifiziert werden.

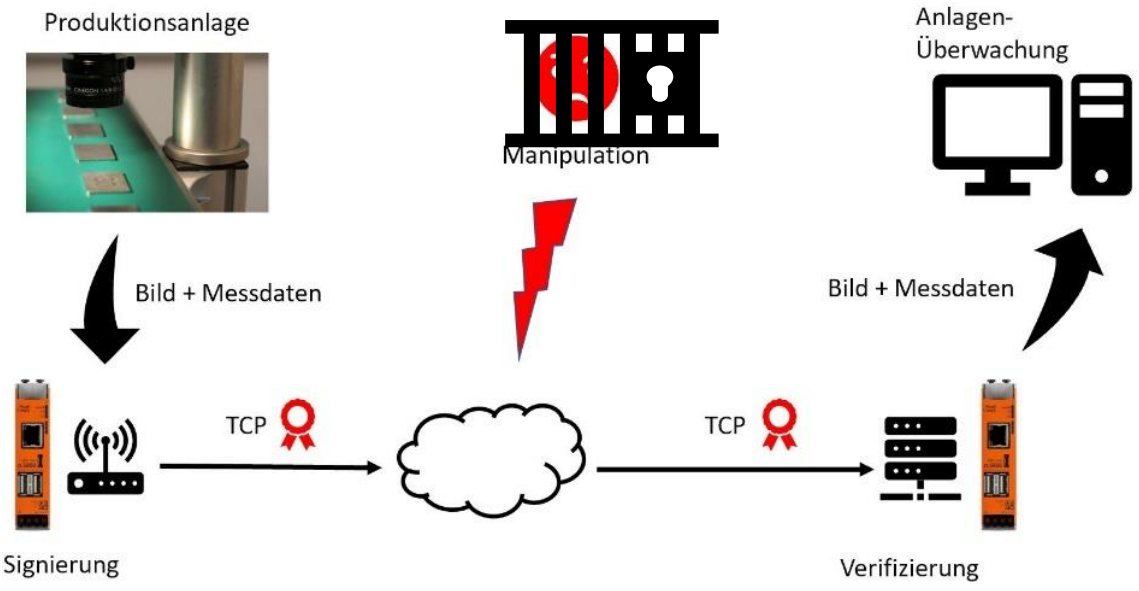

Abbildung 2: Szenario

1.2 Prinzip Darstellung

Das Prinzip des Demonstrators (Abbildung 3) ist in zwei Teile aufgebaut:

- **Senderseite**
  - Simulation der Produktionsanlage
  - Generieren von Daten
  - Signieren der Daten

- **Empfängerseite**
  - Simulation der Anlagenüberwachung
  - Verifizierung der Signatur
  - Plausibilitätsprüfung der Daten

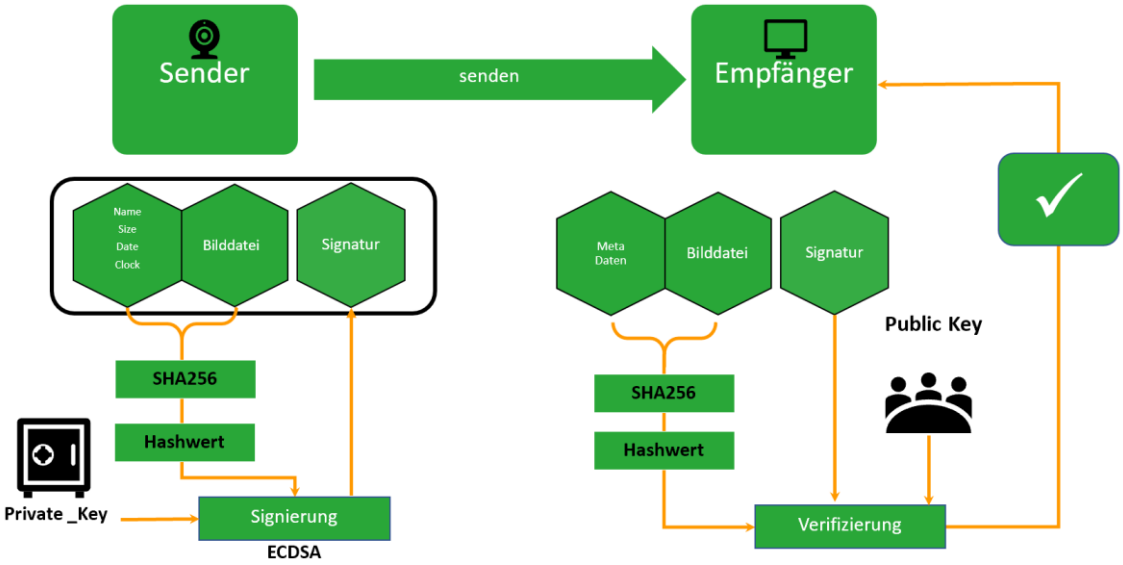

Abbildung 3: Prinzipdarstellung

Die Signierung und Verifizierung erfolgt mittels ECDSA (Eliptic Curve Digital Signature Algorithm) in Verbindung eines SHA256 Hash Algorithmus.

- **ECDSA**
  - Elliptic Curve Digital Signature Algorithm (Asymmetrischer Krypto-Algorithmus)
  - Verwendet Operationen auf Basis elliptischer Kurven
  - Sehr gut geeignet für den Embedded Systems Einsatz
  - Geheimer Schlüssel (private key) zur Erstellung der Signatur
  - Öffentlicher Schlüssel (public key) zum Verifizieren der Signatur

- **SHA256**
  - secure hash algorithm
  - Funktion aus der SHA 2 Familie
  - Beliebig große Eingabemenge (Dateien, Nachrichten, …)
  - Fixe Output Länge von 256-bit = 32 Byte = 64 Zeichen

Auf der **Senderseite** wird eine Nachricht aus den bereitgestellten Daten der Produktionsanlage erstellt. Die Programmsoftware erstellt aus der Nachricht mithilfe des Hashalgorithmus SHA256 einen Hashwert mit 32 Byte Länge. Der generierte Hashwert wird nun mit der ECDSA-Funktion des ATECC508A Security Chips und dem dazugehörigen geheimen Schlüssel signiert. Danach wird die Nachricht inklusive Signatur an die Empfängerseite gesendet.

Die **Empfängerseite** erhält die Nachricht und die Signatur. Auch hier werden die Daten mithilfe desselben SHA256 Algorithmus in einen Hashwert umgewandelt. Da es sich um die gleichen Daten handeln sollte, ist auch der gleiche Ausgabewert der SHA-Funktion zu erwarten. Im Anschluss darauf wird die Nachricht mithilfe des Hashwertes und des im Vorfeld ausgetauschten öffentlichen Schlüssels verifiziert.

Um sicherzustellen, dass das zuletzt empfangene Bild auch nicht von einem Angreifer abgefangen wurde und nun in einer dauerschleife gesendet wird, sind in dem Bild Merkmale wie ein Zeitstempel und ein Zählerwert eingearbeitet. Für diese Werte wird beim Empfänger eine Plausibilitätsprüfung durchgeführt.

Durch diese Maßnahmen werden folgende Schutzziele erfüllt:

- **Integrität**
    - Eine Änderung der Daten ist nicht möglich, ohne bemerkt zu werden

- **Authentizität**
    - Die Quelle der Daten ist sicher auf den Raspberry Pi der Produktionsanlage zurückzuführen

Ein weiterer wichtiger Punkt stellt das Sicherheitsniveau der verwendeten Kryptographischen Funktion dar. Die folgenden Einschätzungen wurde anhand der NIST (National Institute of Standard and Technology) Special Publication 800-57 aus dem Januar 2016 bestimmt [https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-57p1r3.pdf].

- **Sicherheitsniveau des Hashalgorithmus SHA256**
    - Die Sicherheit der Hashfunktion SHA256 liegt bei einer Output Länge von 256-bit bei einer ungefähren Sicherheit von 128-bit
    - Table 3 der NIST Special Publication 800-57
- **Sicherheitsniveau des Signaturalgorithmus ECDSA**
    - Die Sicherheit bei einer ECDSA Funktion liegt ebenfalls bei ca. 128-bit
    - Table 2 der NIST Special Publication 800-57

Somit beträgt das **Sicherheitsniveau** des kompletten Prozesses **durchgehend 128-bit** und gilt nach aktuellem Stand der Technik als ausreichend sicher.

2 Aufbau und Vorbereitung

2.1 Hardwareaufbau

Das Kernelement des Demonstrator Aufbaus bilden zwei Revolution Pi Core-3 [https://revolution.kunbus.de/shop/de/revpi-core-3] der Firma Kunbus. Diese Kontroller sind ab Werk mit einem ATECC508A Security Chip ausgestattet, der zur sicheren Kommunikation in verschiedenen Netzwerken dienen soll.

Die Spannungsversorgung der Geräte erfolgt über die Einspeiseklemmen mittels eines beliebigen industriellen 24V Netzteils. Hier ist es wichtig die geltenden elektrischen Sicherheitsnormen, wie zum Beispiel die DIN-VDE 0100 einzuhalten. Auf Grundlage dieser Norm wurde das 24V-Netzteil mit einem passenden Motorschutzschalter gegen Kurzschluss und Überlast abgesichert. Alternativ könnte hier auch das offizielle Kunbus Stecker Netzteil verwendet werden.

Die Datenkommunikation erfolgt über die Ethernet-Schnittstelle der Geräte und über einen industriellen Ethernet Switch, der ebenfalls mit 24V aus dem Netzteil versorgt wird. Eine drahtlose Kommunikation wäre nur mit zusätzlichen USB-WLAN-Adaptern möglich.

Zugriff auf die Benutzeroberfläche der Revolution Pis erhält man entweder über das Anschließen von Monitor und Eingabegeräten an den vorhandenen Schnittstellen, oder über einen vorher installierten remote Zugriff wie z.B. VNC.

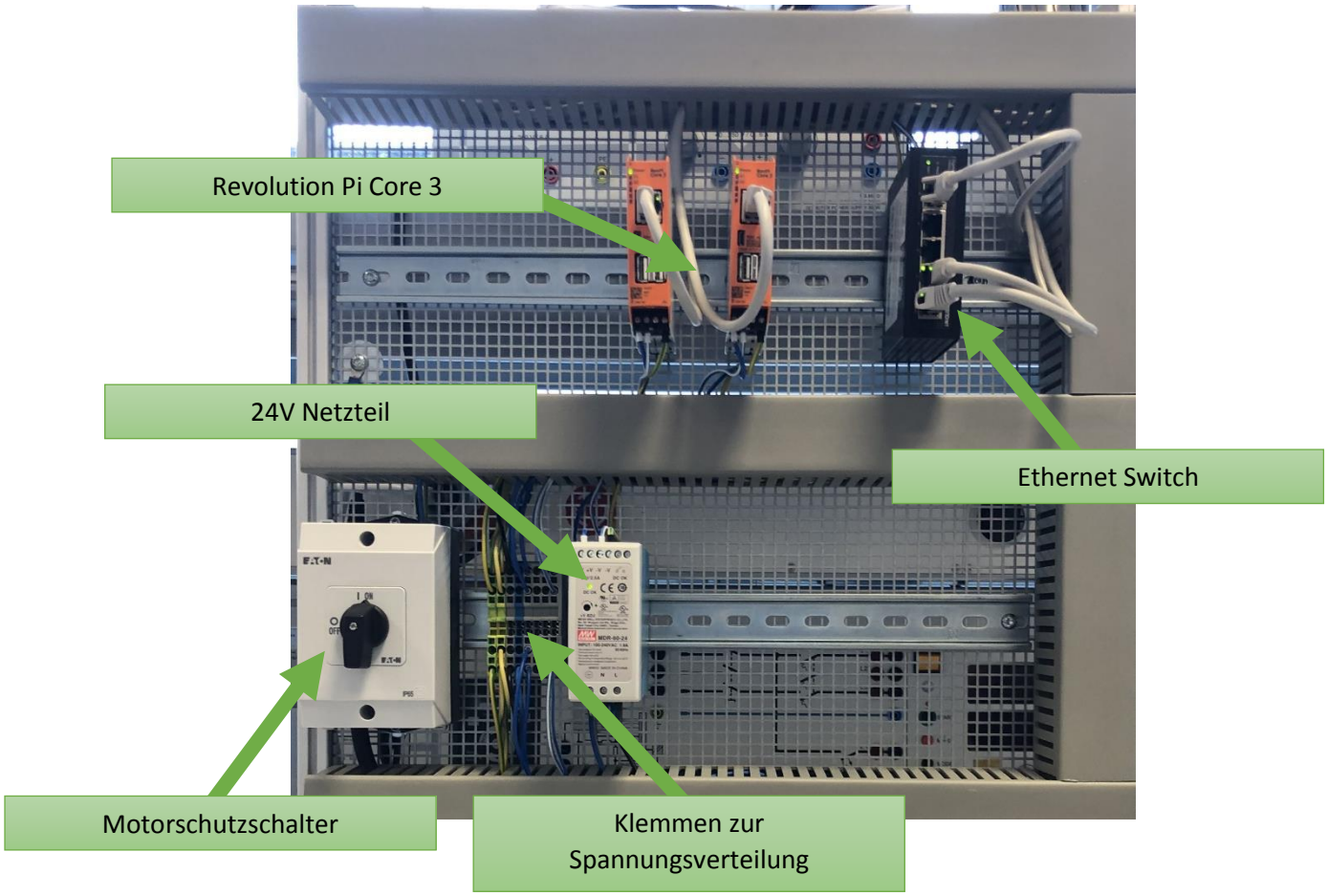

Abbildung 4: Hardwareaufbau

2.2 Softwarepakete

Um das erstellte Projekt für den Signierten Dateitransfer ausführbar zu bekommen, ist es nötig, einige Softwarepakete vorab zu installieren

2.2.1 Revolution Pi

Bevor mit der Installation der notwendigen Softwarepakete begonnen werden kann, ist es unbedingt erforderlich, den Revolution Pi auf den neuesten Stand zu bringen. Hierfür sind folgende Befehle in der Kommandozeile auszuführen:

```
~$ sudo apt-get update
~$ sudo apt-get upgrade
~$ sudo reboot
```

Bei den Punkten 2.2.2 und 2.2.3 kann es zu Fehlern bei der Installation kommen. Dies ist oft auf fehlende Rechte oder Problemen bei den Installationstools zurückzuführen. Bsp*: „Failed building wheel for…“.* Falls diese Fehler auftreten sollten, können folgende Kommandozeilen Befehle eventuell für Abhilfe sorgen:

```
~$ sudo apt-get install build-essential libssl-dev libffi-dev  python-dev libudev-dev
~$ sudo pip3.5 install --upgrade seuptools
```

2.2.2 CryptoAuthLib

Das CryptoAuthLib-Modul (microchip, 2018) ermöglicht den Zugriff auf die meisten Funktionen des ATECC508A Security Chips. Die CryptoAuthLib ist ursprünglich in 'C' geschrieben. Das Modul fungiert daher als Wrapper für die C-Cryptoauth-Bibliotheksfunktionen.

```
~$ sudo pip3 install cryptoauthlib
```

2.2.3 PYQT

PyQt (Riverbank Computing Limited, 2018) ist eine Python-Bindung des plattformübergreifenden GUI-Tool Qt, die als Python-Plug-In implementiert ist. Es beinhaltet alle Werkzeuge für die graphische Benutzeroberfläche.

```
~$ sudo apt-get install python3 python3-pyqt5
```

2.3 ATECC508A

Der ATECC508A Security Chip ist ein Kryptographischer Co-Prozessor mit hardwarebasierter Schlüsselspeicherung. Er ist für Asymmetrische KryptoOperationen optimiert:

- ECDSA: Elliptic Curve Digital Signatur Algorithmus
- ECDH: Elliptic Curve Diffie-Hellman Algorithmus
- NIST Standard P256 Elliptic Curve Unterstützung

Er bietet Speicherplatz für bis zu 16 Keys (je 256-bit). Die Kommunikation erfolgt über den in der Embedded Welt weit verbreiteten I²C-Bus. Es gibt mehrere physikalische Sicherheitsfunktionen zum Schutz vor unbefugtem Zugriff:

- Active Shield Schaltung
- Interne Speicher-Verschlüsselung
- Glitch Protection
- Spannungs-Manipulationserkennung

2.3.1 Beispielkonfiguration

Um den ATECC508A Security Chip benutzen zu können, muss eine Grundkonfiguration auf den Chip aufspielen werden.

ACHTUNG!!!!

Die Konfiguration ist eine One-Way Funktion. Wurde der Chip erst einmal mit den Parametern bespielt und die Konfigurations-Zone geschlossen lässt sich dies nicht mehr rückgängig machen

Um den Chip zu beschreiben, muss die Datei **„crypto_sign_atecc_config.py“** ausgeführt werden. Diese konfiguriert den Security-Chip so, dass mit ihm alle notwendigen Funktionen für den Demonstrator ausgeführt werden können.

Hierbei sei zu erwähnen, dass die Beispielkonfiguration bewusst sehr offen ausgelegt wurde, um nicht andere Funktionalitäten zu blockieren. Im realen Einsatz könnte man hier gegebenen Falls noch Nachbesserungen vorgenommen werden welche die Sicherheit erhöhen.

```
atecc508_config = bytearray.fromhex(
    'C0 00 55 00 8F 20 C4 44 87 20 87 20 8F 0F C4 36'
    '9F 0F 82 20 0F 0F C4 44 0F 0F 0F 0F 0F 0F 0F 0F'
    '0F 0F 0F 0F FF FF FF FF 00 00 00 00 FF FF FF FF'
    '00 00 00 00 FF FF FF FF FF FF FF FF FF FF FF FF'
    'FF FF FF FF 00 00 55 55 FF FF 00 00 00 00 00 00'
    '33 00 1C 00 13 00 13 00 7C 00 1C 00 3C 00 33 00'
    '3C 00 3C 00 3C 00 30 00 3C 00 3C 00 3C 00 30 00')
```

Abbildung 5: Beispielkonfiguration

2.3.2 Excel-Datei als Konfigurationshilfe

Da die Konfiguration des ATECC508A Security Chips sehr komplex ist und eine schier unzählbare Ansammlung von Parametern aufweist, wurde eine Excel Tabelle erstellt, die einen Schritt für Schritt durch die Chipkonfiguration leitet. Genaueres ist der Datei **„ATECC508A Konfig.xls“** selbst zu entnehmen.

3 „Signed File Transfer“

3.1 Programmbeschreibung

Das Programm ist in zwei Teile aufgeteilt, Sender und Empfänger. Das Senderprogramm öffnet zur Demonstration eine Bilddatei, fügt dieser Metadaten hinzu und signiert die ganze Datei mithilfe des ATECC508A. Die Metadaten beinhalten folge Elemente:

- **Seriennummer** des ATECC508A Security Chips der die Signatur erzeugt
- **Zeitstempel** mit Uhrzeit und Datum des Erstellungszeitpunktes der Signatur
- Einen **Zählerwert** der bei jedem neu übertragenen Bild um eins erhöht wird
- Vier zufällige **Sensorwerte**

Anschließend wird die Bilddatei und die Signatur an das Empfängerprogramm geschickt, welches die Signatur verifiziert und eine Plausibilitätsprüfung der Metadaten durchführt.

3.1.1 Sender

Das Senderfenster besteht im Groben aus drei Bereichen:

- Anzeige der aktuellen Bilddatei mit Steuerelementen für den Sendevorgang
- Anzeige der Konfiguration des Security-Chips und Einstellungen des Empfängers
- Anzeige der Metadaten und ECDSA Informationen

3.1.1.1 Anzeige der aktuellen Bilddatei mit Steuerelementen für den Sendevorgang

Unter der Bildanzeige kann durch Drücken des Buttons **„load new pic“** ein neues Bild aus dem Speicher geladen werden. Zum Senden der Bilddateien muss einer der nebenstehenden Sendefunktionen aktiviert werden:

- **„Start Automatic Programm“:**
  - Öffnet ein neues Bild
  - Erzeugt zufällige Sensorwerte
  - Schreibt Metadaten in das Bild
  - Erzeugt eine valide Signatur
  - Sendet Bild und Signatur an den Empfänger
  - Vorgang wird alle 10 Sekunden wiederholt

- **„Send Picture With Invalid Signature”**
  - Schreibt Metadaten in das aktuell geöffnete Bild
  - Erzeugt eine zufällige Invalide Signatur
  - Sendet Bild und invalide Signatur an den Empfänger

- **„Send Picture With Previous Signature”**
  - Schreibt Metadaten in das aktuell geöffnete Bild
  - Verwendet vorher im vorherigen Zyklus erzeugte Signatur
  - Sendet Bild und vorherige Signatur an den Empfänger

- **„Send Picture With Valid Signature”**
  - Schreibt Metadaten in das aktuell geöffnete Bild
  - Erzeugt eine valide Signatur
  - Sendet Bild und Signatur an den Empfänger

3.1.1.2 Anzeige der Konfiguration des Security-Chips und Einstellungen des Empfängers

Die ATECC-Konfigurationsanzeige gibt Aufschluss über die Seriennummer des Chips und dem „Verschlussstatus“ der Konfiguration und Daten-Zone. Zusätzlich wird der die Nummer des Speicherslots angezeigt, der den aktiven private key für die Signatur enthält. Um ein erfolgreiches Senden von Key und Bilddateien zu ermöglichen, muss die IP-Adresse des Empfängers in das Feld **„dest. IP:“** eingetragen werden. Die Ziel-Ports sollten nicht verändert werden.

3.1.1.3 Anzeige der Metadaten und ECDSA Informationen

Die Metadaten, die in die Bilddatei eingefügt werden, können händisch mit beliebigen Werten vorbelegt werden. Eine Ausnahme bildet hier die Seriennummer des ATECC-Chips, welche immer direkt aus dem Chip selbst gezogen wird. Der Zeitstempel kann entweder automatisch aus der Systemzeit bezogen werden oder auch händisch eingetragen werden. Die Auswahl erfolgt über die Checkbox **„use current date and time“.**

Unterhalb der Metadaten befinden sich drei Anzeigefelder, welche Rückschluss auf den erzeugten Hash-Wert, die erzeugte Signatur und den zu verwendenden public key geben. Die Signatur wird nach der Erzeugung nochmal lokal überprüft. Der Hintergrund des Anzeigefenster färbt sich dem Ergebnis entsprechend ein rot oder grün ein.

Zur Generierung eines neuen Schlüsselpaares (private + public pey) kann der Button **„generate new key“** verwendet werden. Der public key muss anschließend dem Empfänger durch klicken des Buttons **„send key“** mitgeteilt werden.

3.1.1.4 Sender Layout

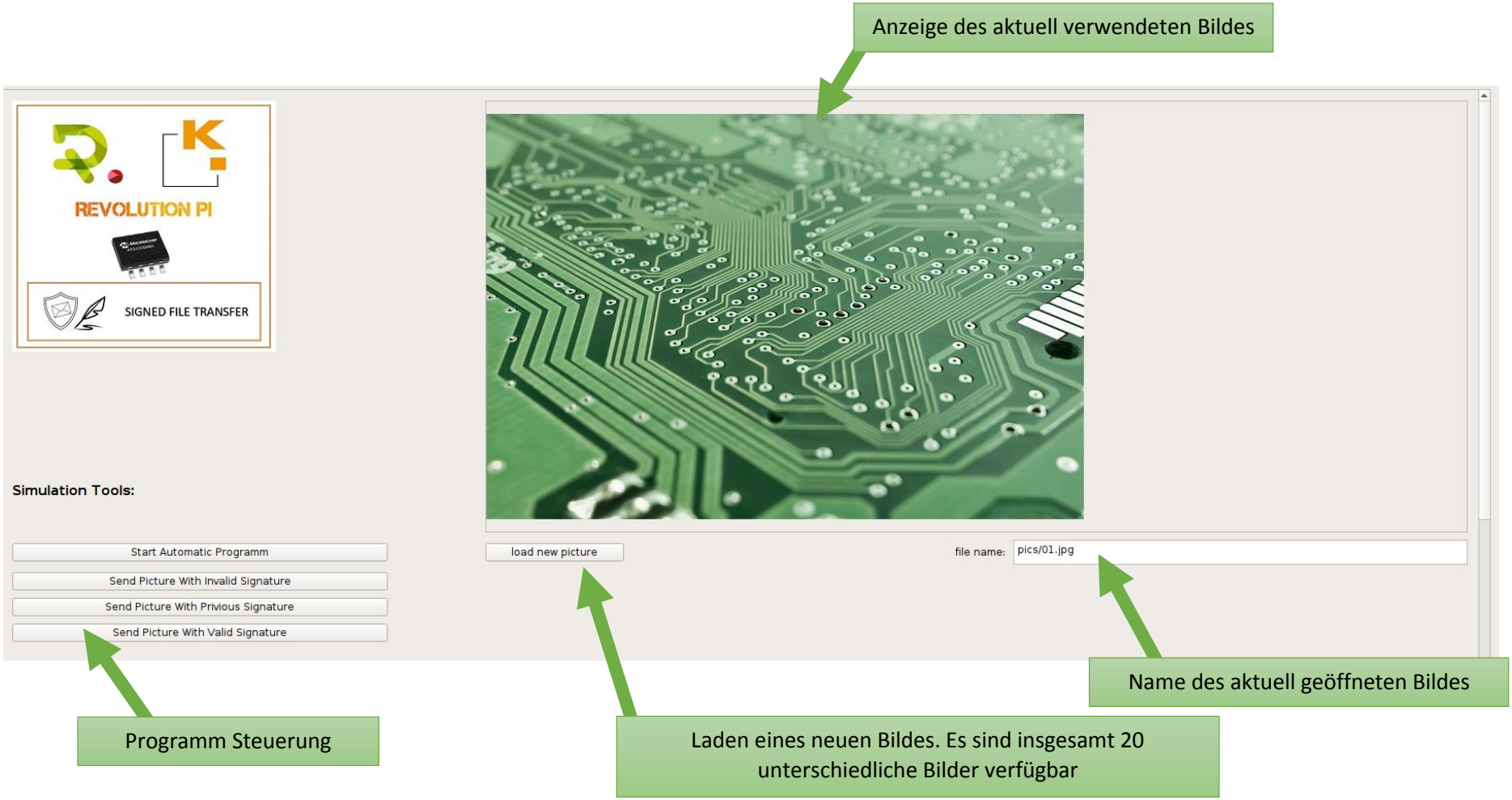

Abbildung 6: Sender Layout Teil 1

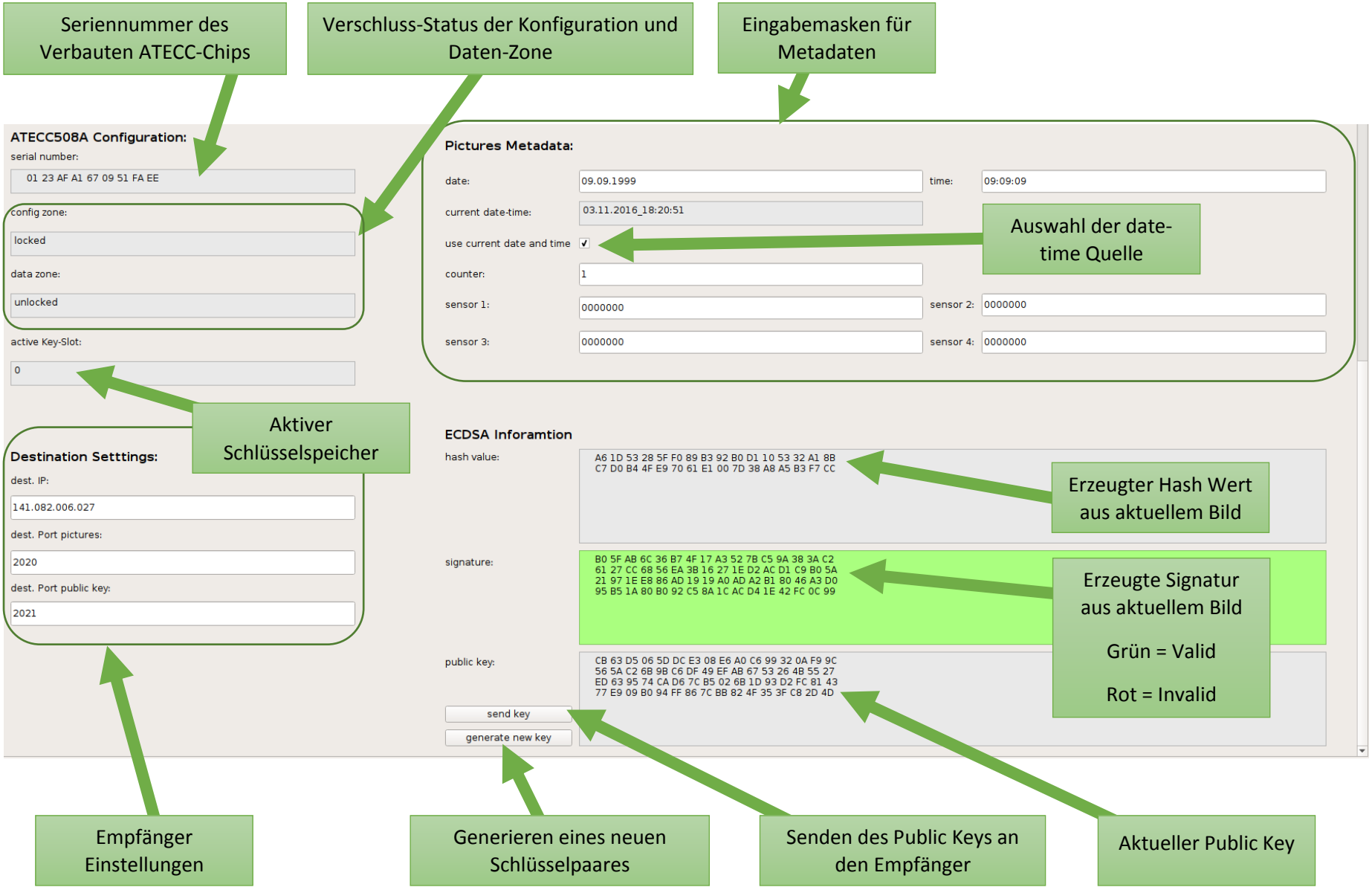

Abbildung 7: Sender Layout Teil 2

3.1.2 Empfänger

Das Empfängerfenster besteht analog zum Sender aus drei ähnlichen Bereichen

- Anzeige der empfangenen Bilddatei mit Validierungsergebnissen
- Anzeige der Konfiguration des Security-Chips und der Kommunikationsschnittstelle
- Anzeige der Metadaten und ECDSA Informationen

3.1.2.1 Anzeige empfangenen Bilddatei mit Validierungsergebnissen

Unter der Bildanzeige kann durch Drücken des Buttons **„save current picture“** das aktuelle Bild abgespeichert werden. Name und Dateipfad erscheinen dann in der danebenstehenden Anzeige. Um neue Bilder empfangen zu können, muss der Button **„Start Automatic Programm“** betätigt werden. Die unter dem Button liegenden Statusfelder beinhalten folgende Informationen.

- **„signature status:“**
  - Validität der empfangen Signatur bezüglich des korrespondierenden Bildes
  - Grün = valide
  - Rot = invalide

- **„counter status:“**
  - Plausibilitätserkennung des mitgesendeten Zählerwertes
  - Plausibel wenn aktueller Zählerwert um eins größer als vorheriger Zählerwert
  - Grün = plausibel
  - Rot = implausibel

- **„date-time status:“**
  - Plausibilitätserkennung des mitgesendeten Zeitstempels
  - Plausibel wenn Zeitstempel innerhalb +/- 60 sec. mit der lokalen Systemzeit übereinstimmt
  - Grün = plausibel
  - Rot = implausibel

3.1.2.2 Anzeige der Konfiguration des SecurityChips und der Kommunikationsschnittstelle

Die ATECC-Konfigurationsanzeige gibt wieder Aufschluss über die Seriennummer des Chips und dem „Verschlussstatus“ der Konfiguration und Daten-Zone. Die „Device Settings“ informieren über die lokale IP-Adresse des Gerätes und die Ziel Ports von public key und Bilddatei.

3.1.2.3 Anzeige der Metadaten und ECDSA Informationen

Das Feld der Metadaten zeigt die im Bild befindlichen Informationen an. Zusätzlich gibt es noch ein Feld, dass den vorherigen Zählerwert wiedergibt.

Unterhalb der Metadaten befinden sich wieder drei Anzeigefelder, welche Rückschluss auf den erzeugten Hash-Wert, die empfangene Signatur und den verwendenden public key für die Verifizierung geben. Wenn ein neuer Key empfangen wird, erscheint dieser mit einem orangen Hintergrund. Dies ist ein Indikator dafür, dass zwar ein neuer Key empfangen wurde, dieser jedoch noch nicht abgespeichert wurde und somit auch nicht verwendet wird. Das aktivieren des public keys muss manuell über den Button **„save key“** erfolgen.

3.1.2.4 Empfänger Layout

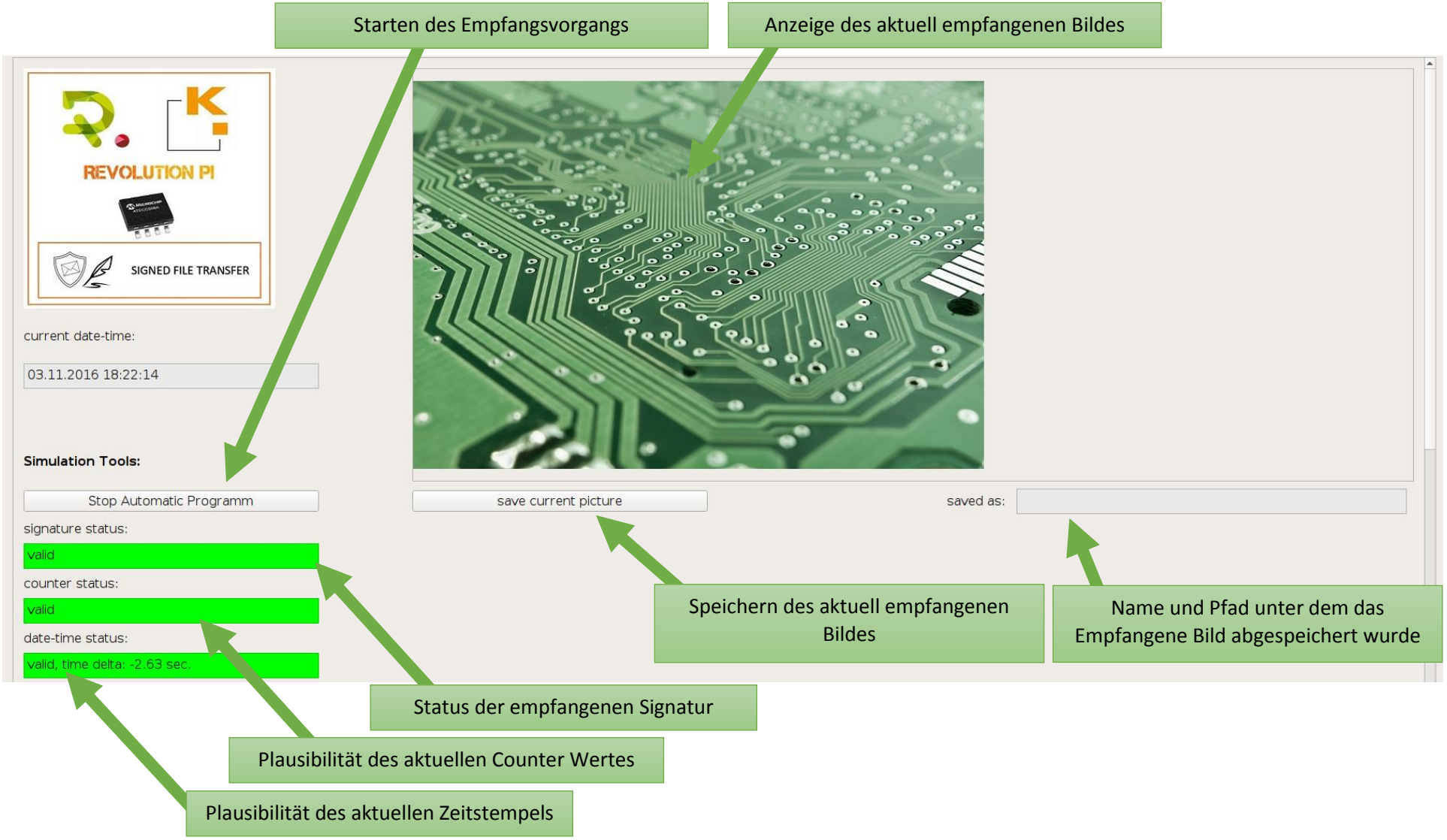

Abbildung 8: Empfänger Layout Teil 1

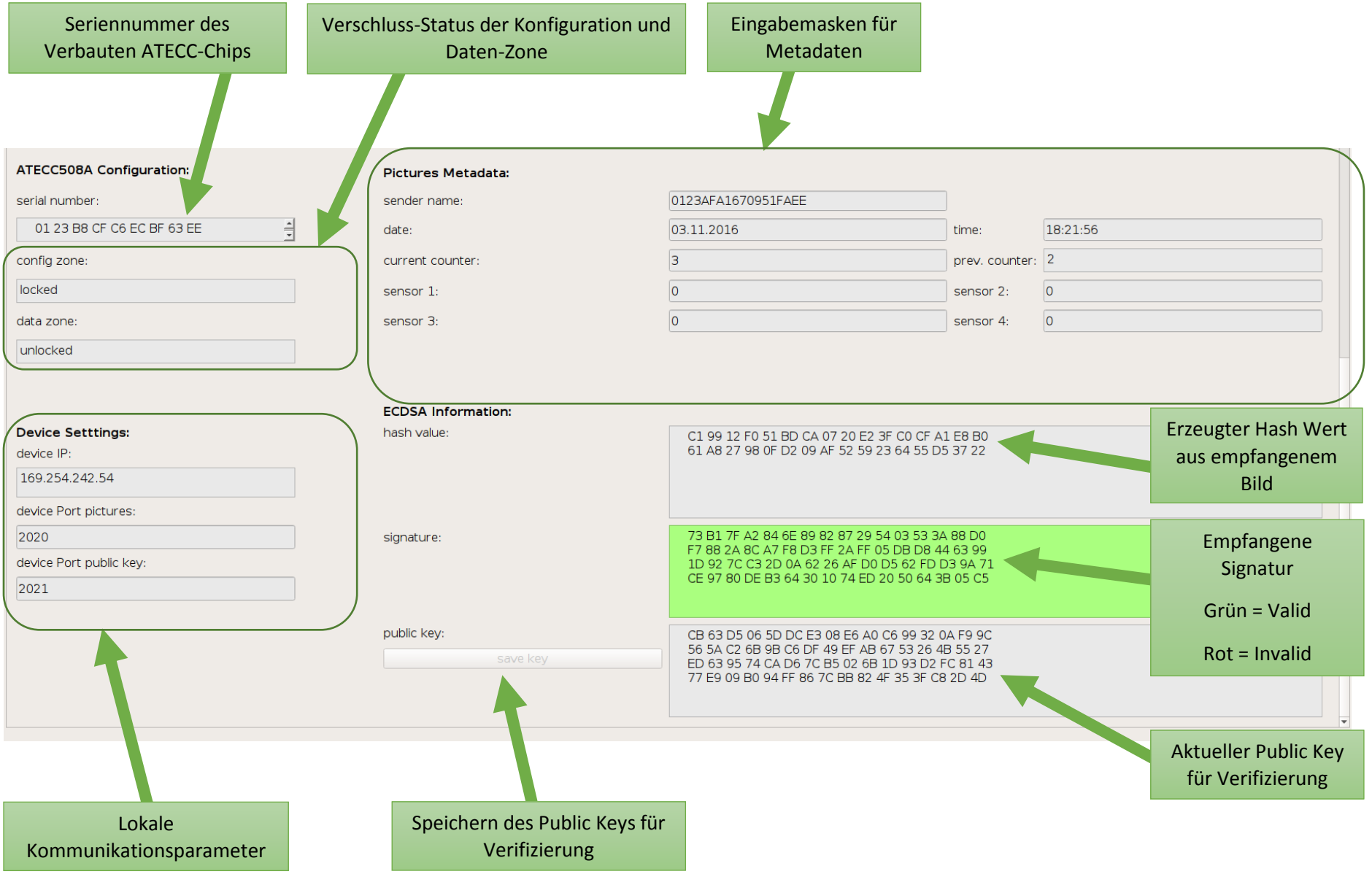

Abbildung 9: Empfänger Layout Teil 2

3.2 Softwarestruktur

Dies soll einen kleinen überblick über die verwendeten Funktionen geben. Genauere Informationen sind den Kommentaren des Quellcodes zu entnehmen.

Es sind folgende Dateien für den Betrieb des Demonstrators notwendig:

- pics/01.jpg bis pics/20.jpg (pixabay, 2018)
- logo.jpg
- common.py (microchip, 2018)
- crypto_sign_atecc_config.py
- send_design.py
- reveice_design.py
- send_main.py
- receive_main.py

3.2.1 Sender

Hier eine Auflistung der Funktionen in der send_main.py

- __init__
- click_new_pic
- click_send_valid
- click_send_valid_auto
- click_send_invalid
- click_send_invalid_prev
- click_gen_new_key
- click_send_pub_key
- chgtime
- setip
- setport_pic
- setsen1
- setsen2
- setsen3
- setsen4
- setmandatetime
- timer_update
- set_metadata
- TCP_send_pic
- TCP_send_key
- init_device
- closeEvent

3.2.2 Empfänger

Hier eine Auflistung der Funktionen in der receive_main.py

- __init__
- click_save_pic
- click_start_auto
- click_save_pub_key
- setport_pic
- setport_pubkey
- timer_update
- get_metadata
- init_device
- get_ip
- TCP_receive_pic
- TCP_receive_key
- verify
- closeEvent

4 Fazit

Der Revolution Pi Core3 der Firma Kunbus bietet in Zusammenarbeit mit dem ATECC08A Security Chip eine gute Kombination für die sichere Kommunikation von Maschinen. Hierdurch konnte im Demonstrator Aufbau ein Integritäts- und Authentizitätsgeschützter Kommunikationskanal geschaffen werden.

Die Programmoberfläche von Sender und Empfänger geben einen Überblick über die Funktionsweise des signierten Datentransfers und erlauben die realistische Simulation möglicher Manipulationsversuche und wie diese erkannt werden können.

5 Abbildungsverzeichnis

6 Literaturverzeichnis

microchip. (25. Januar 2018). *cryptoathlib*. Von github: https://github.com/MicrochipTech/cryptoauthlib abgerufen

microchip. (6. November 2018). *cryptoauthtools*. Von github: https://github.com/MicrochipTech/cryptoauthtools abgerufen

pixabay. (17. Dezember 2018). *Hans Braxmeier & Simon Steinberger GbR*. Von https://pixabay.com/ abgerufen

Riverbank Computing Limited. (3. Oktober 2018). *pyqt5*. Von pypi: https://pypi.org/project/PyQt5/ abgerufen