Systembeschreibung

Christoph Bieringer, Simon Schneider

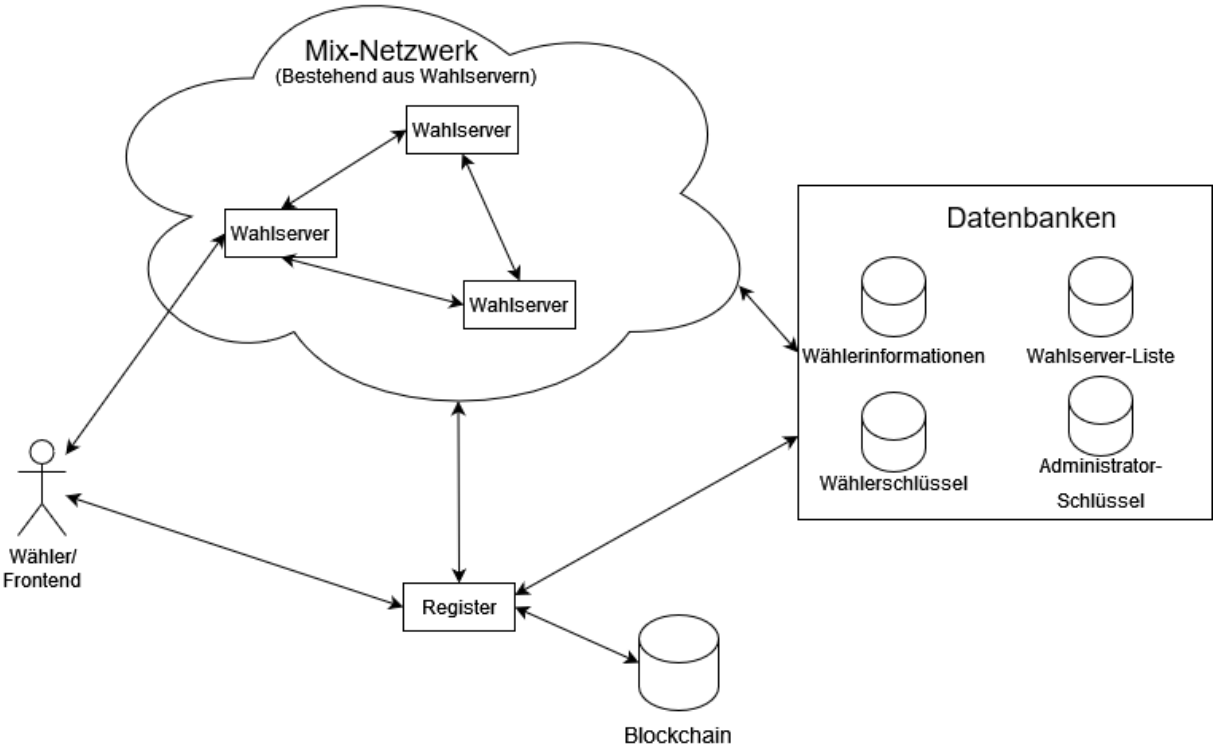
26.10.2022

Da die Dokumentation der Vorgängergruppe tlw. recht spärlich war, wurde beschlossen, zu Beginn der Arbeit eine Beschreibung des gesamten bisher implementierten Systems zu erstellen. Dieses Dokument enthält die Ergebnisse. Eine Beschreibung der im Rahmen dieser Arbeit vorgenommenen Änderungen (und damit des aktuellen Systems) befindet sich im Dokument „Vorgenommene Änderungen“.

Inhalt

Übersicht

Eine Übersicht über das gesamte bisher implementierte System bietet das folgende Diagramm.



Der Benutzer interagiert mit dem Wahlsystem im Wesentlichen nur über das Frontend, eine Website. Wenn er über dieses seine Stimme abschickt, wird diese an einen Wahlserver geschickt und von diesem verschlüsselt (tatsächlich verschlüsselt wird im aktuellen System ein String mit dem Namen der Partei). Es gibt mehrere fast identische Wahlserver[1], die zusammen (ansatzweise) ein Mix-Netzwerk bilden. In diesem wird jede abgegebene Stimme mehrmals von Server zu Server geschickt, bevor sie schließlich am Ende an das sog. Register, einen weiteren Server, geschickt wird. Das Register verwaltet eine Blockchain, in der es die angekommenen Stimmen abspeichert. Diese können auch wieder ausgelesen und entschlüsselt werden (mit dem privaten Schlüssel des Wählers). Auf diese Weise können einzelne Wähler verifizieren, dass ihre Stimme korrekt gespeichert wurde. Ebenso können alle Stimmen auf einmal ausgelesen und ausgezählt werden, um das Wahlergebnis zu berechnen (hierfür hinterlegen die Wähler eine verschlüsselte Version ihres privaten Schlüssels, die bei Bedarf mit einem Administratorschlüssel entschlüsselt werden kann). Für diese beiden Funktionen wiederum steht im Frontend eine passende GUI zur Verfügung.

Im Folgenden werden zunächst die einzelnen Komponenten des Systems genauer betrachtet. Anschließend werden die Abläufe im System vor einer Wahl, bei der Stimmabgabe, bei der Verifikation und bei der Auszählung noch genauer beschrieben.

[1] In der aktuellen Implementierung unterscheiden sich die Wahlserver nur dadurch, auf welchen TCP-Port sie hören.

Komponenten

Frontend

Beim Frontend handelt es sich um eine einfache Website, die in HTML mit CSS-Stylesheets und JavaScript als Skriptsprache geschrieben wurde. Die entsprechenden Dateien befinden sich im Ordner „dhbw-blockchain-website“. Es gibt drei verschiedene, untereinander verlinkte Pages für folgende Funktionen:

- Stimmabgabe (in der Datei index.html): Hier kann der Wähler nach Eingabe der erforderlichen Authentifizierungsdaten aus einem Menü seine Wahl treffen und abschicken.
- Verifikation (in „index2.html“): Hier kann der Wähler nach Eingabe verschiedener Informationen seine Stimme aus der Blockchain auslesen und entschlüsseln lassen, um sicherzugehen, dass sie korrekt abgespeichert wurde.
- Auszählung (in „index3.html“): Hier kann der Wähler alle Stimmen aus der Blockchain auslesen, entschlüsseln und auszählen lassen. Das so errechnete Wahlergebnis (bzw. der aktuelle Zwischenstand) wird ihm dann in einem Kreisdiagramm (erstellt mithilfe der anychart-Bibliothek) angezeigt.

Wahlserver

In der aktuellen Lösung werden drei nahezu identische Wahlserver verwendet. Sie wurden in Python und mithilfe des Flask-Webframeworks geschrieben. Ihr Code befindet sich im Ordner „dhbw-blockchain-encryption“ in den Dateien „main.py“, „main1.py“ und „main2.py“.

Sie bieten in einem REST-artigen Interface mehre API-Endpoints an.

- Über /api/transmit können Wähler ihre Authentifizierungsinformationen vorzeigen und eine Stimme abgeben.
- Über /api/getAuszaehlung kann eine Auszählung (d.h. die Anzahl an Stimmen für jede Partei) erhalten werden.
- Über /api/receive empfangen Knoten innerhalb des Mix-Netzwerks Nachrichten (bestehend aus einem persönlichen Hash und einer verschlüsselten Stimme) von ihren Vorgängern und senden diese dann an den nächsten Knoten weiter

Register

Das Register ist analog zu den Wahlservern ein in Python geschriebener, Flask-basierter Server. Sein Code befindet sich in „dhbw-blockchain-encryption/register.py“.

Auch es bietet mehrere Endpoints an.

- Über /api/commitVote nimmt es Stimmen (von einem Wahlserver (dem Exit-Node des Mix-Netzwerks)) entgegen (eine Stimme besteht dabei, wie oben beschrieben, aus einem persönlichen Hash und einer verschlüsselten Stimme).
- Über /api/getTransactions können alle bisher angekommenen, verschlüsselten Stimmen ausgelesen werden.
- Über /api/getVote kann eine Stimme (über den persönlichen Hash) ausgewählt und mit einem vom Client bereitgestellten privaten Wählerschlüssel entschlüsselt werden.

Die ankommenden Stimmen bestehen aus einem persönlichen Hash zur Identifizierung und der eigentlichen, mit dem privaten Wählerschlüssel verschlüsselten Stimme. Sie werden in einer Blockchain gespeichert, wobei jede Stimme (persönlicher Hash und Stimme) einen Block ausfüllt. Jeder Block enthält dabei zusätzlich einen Integritätshash, der aus den Blockinhalten und dem

Integritätshash des vorigen Blocks besteht. Auf diese Weise hängt der Integritätshash eines Blocks von allen vorigen Blöcken ab. Diese können nicht mehr rückwirkend verändert werden, ohne die Integritätshashes in den späteren Blöcken fehlerhaft zu machen. Auf diese Weise erreicht die Blockchain ein hohes Maß an Manipulationssicherheit.

Datenbanken

Das System verwendet mehrere Tabellen, die derzeit alle in einer einzelnen MariaDB-Datenbank gespeichert werden. Es gibt folgende Tabellen:

- voters: Speichert für jeden Benutzer eine persönliche ID, einen Authentifizierungscode und ob er schon gewählt hat.
- server: Speichert eine Liste von Wahlservern mit den passenden URLs.
- directory_server3: Speichert eine Liste von persönlichen Hashes mit den dazugehörigen verschlüsselten Wählerschlüsseln.
- admin_key_server: Speichert den Administratorschlüssel, der für die Entschlüsselung der Wählerschlüssel verwendet wird.

Bibliotheken

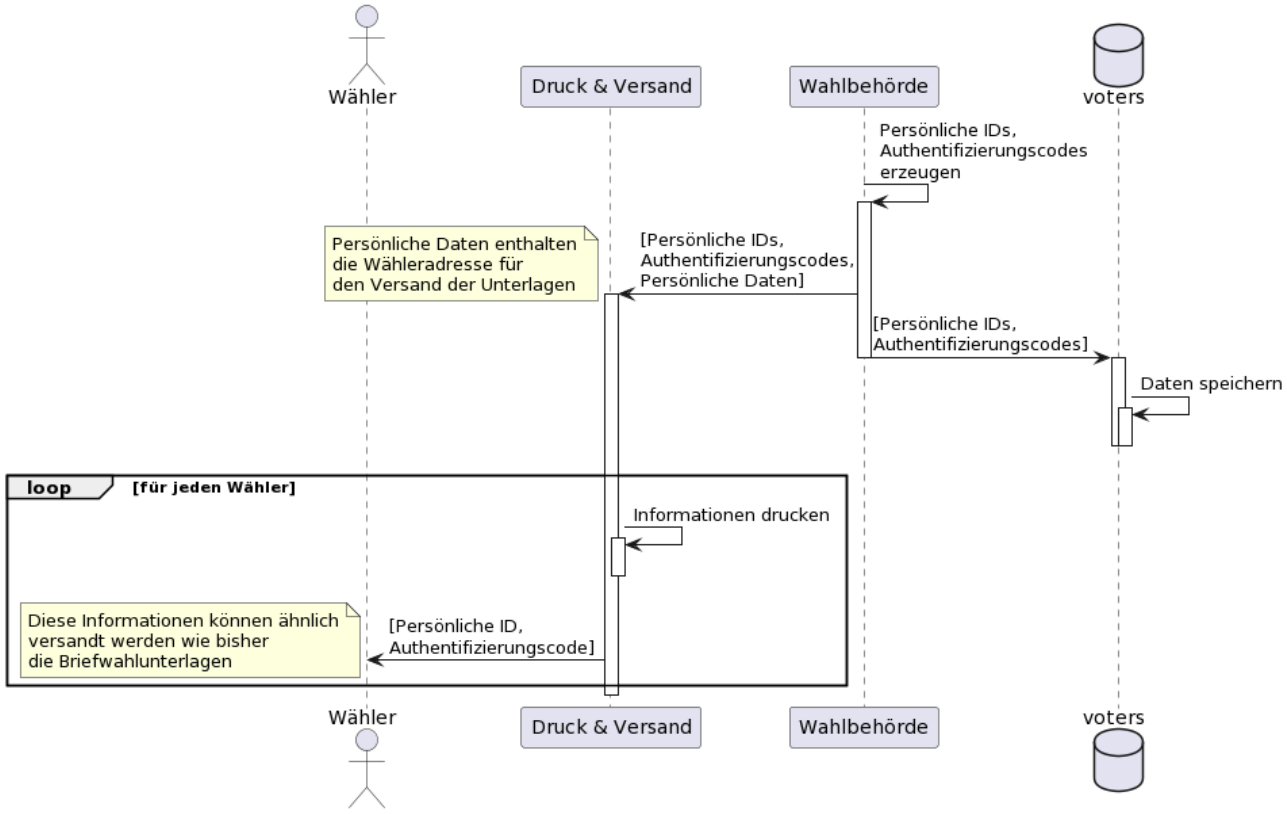
Wahlserver und Register verwenden für ihre Arbeit mehrere in Python geschriebene Bibliotheken. Diese liegen in „dhbw-blockchain-encryption“.

- auth_service.py enthält verschiedene Funktionen zum Zugriff auf die Datenbank und zur Benutzerauthentifizierung.
- encryption_service.py enthält verschiedene Funktionen zum Ver- und Entschlüsseln sowie zur Schlüsselerzeugung und zum Erstellen von QR-Codes.
- blockchain.py enthält eine Implementierung der weiter oben beschriebenen Blockchain in einer Klasse.

Vor der Wahl

Vor der Wahl müssen die benötigten Authentifizierungsinformationen, bestehend aus einer Persönlichen ID und einem Authentifizierungscode, erzeugt und an die Wahlberechtigten verteilt werden. Dieser Schritt wurde von der Vorgängergruppe nicht behandelt. Das folgende Diagramm stellt daher nur einen möglichen Ablauf da.

Die zuständige Behörde erzeugt die Authentifizierungsinformationen für alle Wähler und lässt diese von einem Dienstleiter drucken und (auf dem Postweg) ausliefern. Parallel werden die Informationen in der voters-Tabelle gespeichert, damit sich die Wähler später authentifizieren können.



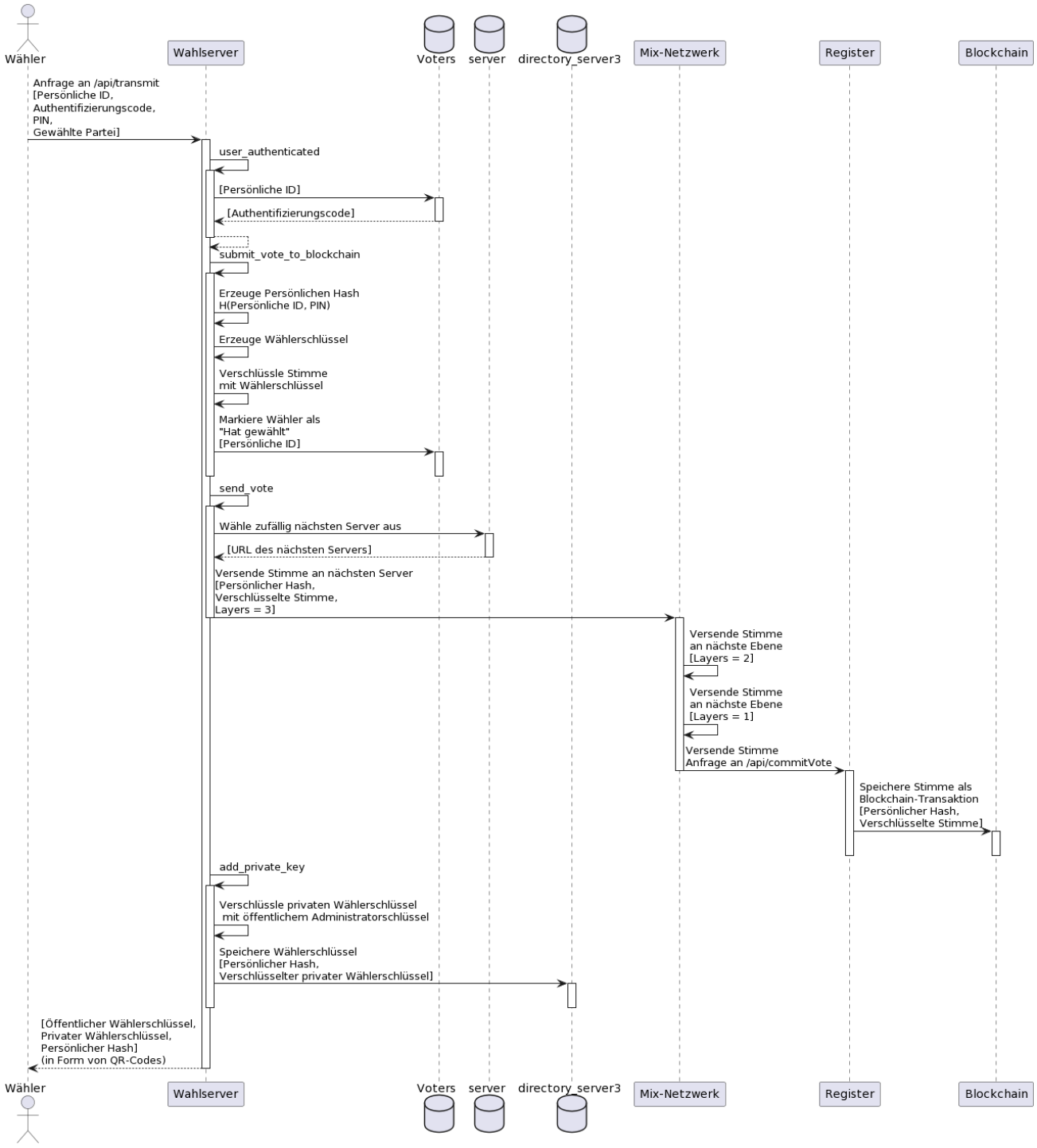
Ablauf der Stimmabgabe

Bei der Stimmabgabe sendet der Wähler/das Frontend seine Stimme zusammen mit seinen Authentifizierungsdaten und einer von ihm gewählten PIN an einen der Wahlserver. Dieser überprüft zunächst mithilfe der voters-Datenbank, ob der Wähler überhaupt wahlberechtigt ist. Anschließend wird aus Persönlicher ID und PIN ein persönlicher Hash-Wert errechnet, der im Folgenden zur Identifikation der Stimme genutzt wird. Der Wahlserver erzeugt zudem ein Schlüsselpaar für den Wähler. Die neu erzeugten Informationen werden als QR-Codes abgespeichert. Der Wahlserver markiert nun den Wähler in der voters-Tabelle als „hat gewählt“.

Die eigentliche Stimme (d.h. ein String mit dem Namen der gewählten Partei) wird mit dem privaten Schlüssel des Wählers verschlüsselt und zusammen mit dem persönlichen Hash in das Mix-Netzwerk gesendet. Der nächste Server wird dabei zufällig aus der server-Tabelle ausgewählt. Nach mehreren (aktuell 3) Layern wird sie schließlich an das Register gesendet und in der Blockchain (als [Persönlicher Hash, Verschlüsselte Stimme]-Paar) gespeichert.

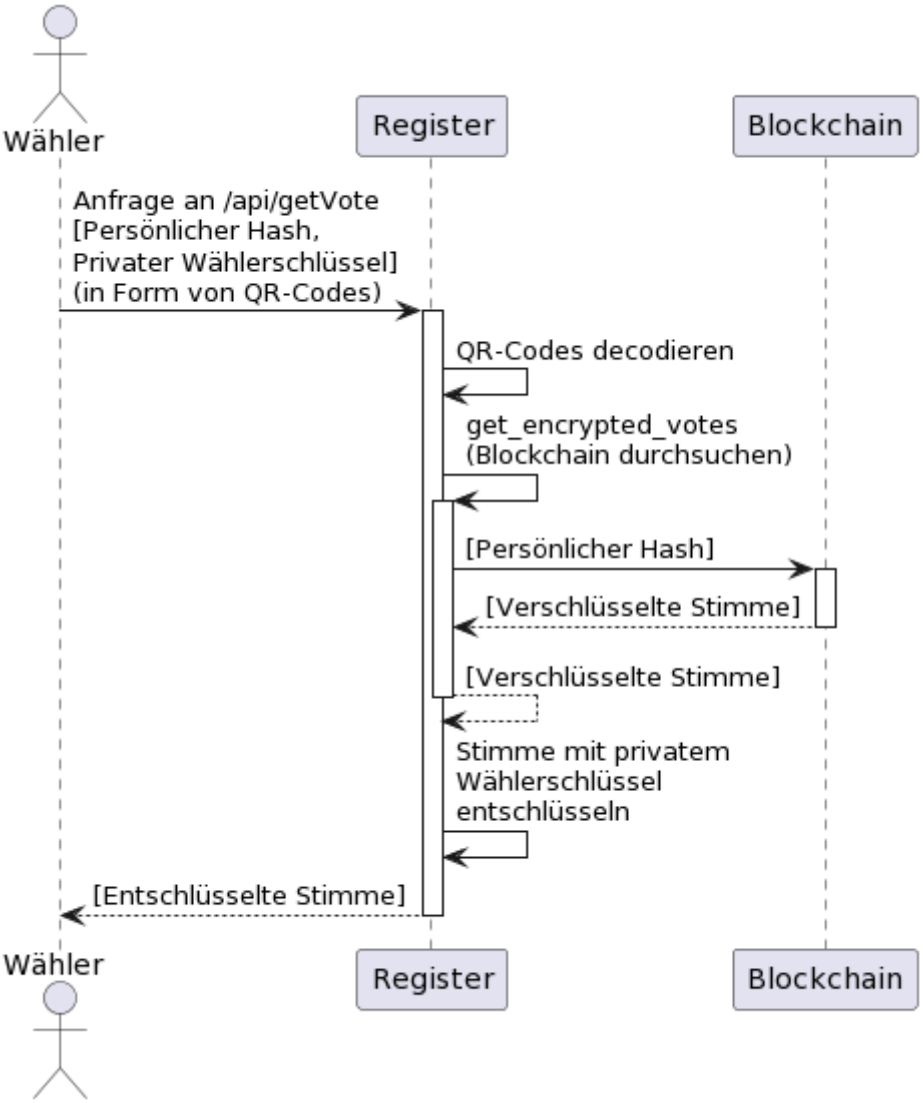
Der private Wählerschlüssel wird nun noch mit einem sog. Administratorschlüssel (genauer gesagt dem öffentlichen Teil des Administrator-Schlüsselpaars) verschlüsselt und zur späteren Verwendung in directory_server3 gespeichert.

Abschließend erhält der Wähler die erzeugten Informationen (Persönlicher Hash und Schlüsselpaar) in Form von QR-Codes für die spätere Verwendung.



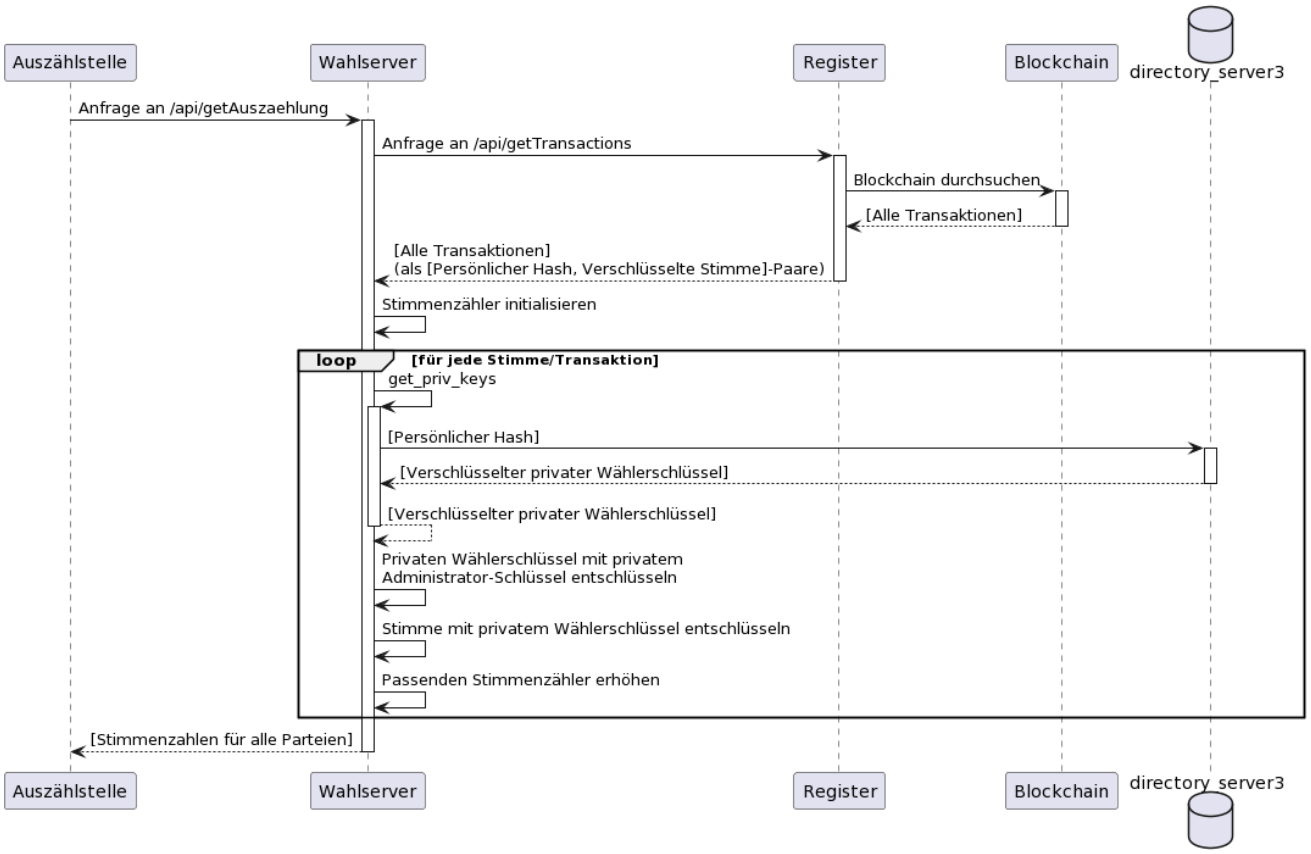
Ablauf der Verifikation

Zur Verifikation der Stimmabgabe stellt der Wähler eine Anfrage an den entsprechenden API-Endpoint des Registers. Die Anfrage enthält dabei als Daten den persönlichen Hash des Benutzers sowie seinen privaten Schlüssel (beides in Forme von QR-Codes). Das Register sucht die entsprechende Stimme aus der Blockchain heraus, entschlüsselt sie mit dem privaten Wählerschlüssel und übermittelt das Ergebnis an den Wähler. So kann sich dieser davon überzeugen, dass seine Stimme korrekt verarbeitet wurde.



Ablauf der Auszählung

Für eine Auszählung wird eine Anfrage an den entsprechenden API-Endpoint eines Wahlservers gestellt. Dieser holt sich daraufhin alle Einträge aus der Blockchain des Registers. Diese entschlüsselt er nun, wobei die hierfür notwendigen (zunächst noch verschlüsselten) Wählerschlüssel aus directory_server3 geholt und mit dem Administratorschlüssel entschlüsselt werden. Die persönliche Hashwert dient dabei als Bindeglied für die Zuordnung zwischen Stimme und Wählerschlüssel. Nach dem Entschlüsseln einer Stimme wird ein entsprechender Zähler erhöht. Nachdem alle Stimmen auf diese Weise ausgewertet wurden, werden die Zählerstände an den Nutzer zurückgeschickt.



Zur verwendeten Kryptographie

Für die hier beschrieben Ver- und Entschlüsselungsvorgänge kommt Elliptic Curve Cryptography (kurz ECC), d.h. ein asymmetrisches Verschlüsselungsverfahren, zum Einsatz. Sowohl Wähler- als auch Administratorschlüssel haben also einen öffentlichen und einen privaten Teil, die jeweils zum Verschlüsseln und zum Entschlüsseln benötigt werden.

> Detail: Tatsächlich wird für das Ver- und Entschlüsseln die eciespy-Bibliothek (kurz für *Elliptic Curve Integrated Encryption Scheme for secp256k1 in Python*) verwendet. Diese verwendet Elliptic-Curve-Diffie-Hellmann über der (auch von verschiedenen Kryptowährungen verwendeten) Kurve secp256k1, um einen AES-Sitzungsschlüssel sicher auszutauschen. Die eigentliche Ver-/Entschlüsselung der Nutzdaten findet dann mit 256-Bit-AES statt. Für weitere Details sei auf die entsprechende Dokumentation (https://pypi.org/project/eciespy/) verwiesen.

Die persönlichen Hashes der Wähler werden mit der argon2-Hashfunktion erzeugt. Für die Integritätshashes der Blockchain wird SHA256 verwendet.

Die gesamte kryptographische Aktivität findet serverseitig, d.h. in Python-Code, statt. Die Implementierungen der kryptographischen Primitive entstammen den Paketen eciespy (ECDH + AES), argon2-cffi (argon2-Hashfunktion) sowie hashlib (SHA256-Hashfunktion).

Zu den UML-Grafiken

Die in diesem Dokument verwendeten Sequenzdiagramme wurden mit PlantUML, einem webbasierten Tool (verfügbar unter https://plantuml.com), erstellt. Der Quellcode zu den vier Diagrammen ist in den folgenden vier URLs codiert:

<u>Vor der Wahl:</u>

//www.plantuml.com/plantuml/png/bLCnhjf04Ett52EALE4250WYDtGq2KLCCNZ7UCCojcRNHk8yEGD
LEbyiirWMDaca_CexCs-VtdlZtOI84jkhDziYrWA7_cVbIG03T6Ql8rUu1Zd28fEskC0d-
4OIaEqYGHexnnomyYUg-
eTOUW84JOOHJnW8kZhgrTIrdJ62oxNW2lRQwn_iNL4HxBBmEUTj6olYw4ftToGjdqDHMmevazofFHCR
BABhC1889YPOhbNA2hx_FvbEPc1LINny4QipLhtWgOmh-
1CAobcXrrFErj6O6rgXe5xB_fVeW7YC2om9V6LDm6DoC8YstOT47usbgVNnd0ArtKWuw0iD9KBXEOgcr
NlfwxfvIeSVfDf7KmP0ynX3td5PonMZgraXDlqLwWS0fjbfQUcjCL3ynV4_3H_JjSaQCaU1tWHSb2XzLUgJ
CETkI3p2ZIIvkEbwJYve5iEgleYZyeQLRr_h8BR6p1LljAVluZS0

Stimmabgabe:

//www.plantuml.com/plantuml/png/fLJ1Jjj04BtxAqOvWTG8jRb5QaLQQ442HA650xQipUx4NiKUHxjh
0_cGVW7VmAat_rXt4oVhn6xLgeSWrczctfktCtki3LEcpBDWmasXuBv-
IJDKm3HKmRM599SBHWRkMPffL5M3wU246JPb6k6kCAYq-
oeg_zN72lUl1uHKwAYUutN8Q1qpsc4RNCgduHMQvIEg-S15vF9f9-864wdDca5XieDzp0e-
vocJvD0f3u8AXX-
iw35CQAPOWi08ZjX27Xd5IEVIXFHmZKhNhvH9dYASdxm9QLAQ5Cd8cLnAL2KbcXS2BN1zVcN_Vi57vvP
1SEGeeo1WtCYA6RHSbbHlM4kDAcPDDMvnqOvi0aKrXdqHaO-
pvYeqfI9uwDSLDNYRM9TJsrnS5GPZKyJJhJ4z_2x-
L2snjFQqPH2SCPs6T7Qmxn1O7mvxifiXqZpDwfzQOzQAkVFVCuHR8_CSGx8YUxAyBvTCpIKgzFEPwP06P
yn0qjp28AG_koUmS-
gxAuqaLePrBLgFyhWH0SjoLhzacKo0wXUUQeC4jytObzgdkon_UTzlBa2WxkJeg26naxws2Kbi7096dOIzV
jNgcknOthKzD9iA5hXWpnQ2zp2AzjlV7joQsT_lT8JK4WAdKoIqChQ5tqN_aVnscwmmWPxi_TtbHMxdnZ
q_xNPi3T-FAnNOdwiwvMEuNU3AA--
hcoF_NWo_kVU0pOqiwD_yzRDcYRmUV_9wMiF6X8WNQWN5Stpk6RdkpZJnr5sOaC0jKlqwco6PbUeS
9YAN9BLHp1GzowL7RLSw9V_I1BLHfRebl3EEpPlXZxxdxPRKFtmFFJNT6zmbz61NSnJIWIJuNAWSgeBW
wytmKo5G7mR7IABCiry0

Verifikation:

//www.plantuml.com/plantuml/png/ZP0zJWCn48Lxds9Aa8ZIfq011A8CGGf5DegSyR2siCUhyMmYxiCP
gEXoCNQ9o8z8GU72_iRwtfjnKIlQfcWiQXPusNxuI0AsmCOq7GeO6iiAKwf3qHqHga_ORSpuXju6xkaAZT
d0uBeV6y4Dlugj2Ip3q3PXM9FEibB5ymb9sNvo3EX9uD4MVrNnHCBQAks35FHn-
rKAnKN55vtW8Kk2TMPucWxkigDosIV_-TGBJNT-
tViHS8JP1HBY4rwJBebHtXibjrndfT99ZjguLj2N5ZtnUSqAHt2coE8ml48ZfAqmp6V7TJhqh24bMlo37oBlt
Y05XMQtgbJndrrLJAo7A-yLzwnd5MDYrwRu3G00

Auszählung:

//www.plantuml.com/plantuml/png/dPAnRjj038PtFGN75gZX8Pk78YuwT1IQeXaImw1FZCJwH1aa9I1z
dZv3fcnwiTvLS6J7GGjqC63W__vVFycxDaVrler5YUeS-83YiExjv_YhYUOK8m4QE9u1jvX4qe4qYuETYL-
fPlD9KghFj4-n2_lG84jMTw7uZ8uxD8AAbO9t-hYTWgyoKDbLKJZ2ucFwo0hMyg1O4w304W-
yhCbpLQGczb8N69m7TCg51iicfVfDrpT5iOntOhCjbqs_RDo5rMdNgjVGM1yQaXbFxPMyLu6xTLxJ5BhFe
IIRlsdtyWwZmLr9QkEJH4x12b_GcW_mdTH24yTdCufESEFSjhHPb8XAxuiqs74Rao9_BfNiBEoCaOr9IOeY
Tjq17iPdXHzK7NEM9qLEishoxK5vsExfqIuNcG-
nUgllPcOJyZBpgo4KSZRcFxTJsrdT_9zfhbvEb5nIqB8VGzfxMLSj2vihUgUBcnaZyTaa5ni-
Yvc_yczxYMOa5SdhKv4surD-IqajNkO_GfWOcIw7CLAAKIVEJ-ewyNqRVmC0

Beim Zugriff direkt auf diese URLs wird von plantuml.com das entsprechende Bild erzeugt und im Browser dargestellt. Der Quellcode zu einer URL lässt sich im sog. „Online Editor“ (auf der PlantUML-Homepage in der Navigationsleiste verfügbar) einsehen. Hierzu wird die URL im Editor in das entsprechende Eingabefeld kopiert und mit dem Button „Decode URL“ decodiert.