Analyse und Implementierung einer Multi-Faktor-Authentifizierung mit Shamir's Secret Sharing

Nicolas Proske
Ostbayerische Technische Hochschule Amberg-Weiden
Moderne Anwendungen der Kryptographie
E-Mail: n.proske@oth-aw.de
Matr.-Nr.: 87672270

***Zusammenfassung*—Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.**

***Schlüsselwörter*—MFA, Secret Sharing**

I. Einleitung

In der heutigen digitalen Welt, in der eine überwältigende Menge an Daten generiert, gespeichert und über verschiedene Plattformen übertragen wird, ist der Bedarf an Datenschutz und Datensicherheit relevanter denn je. Die mit der Digitalisierung einhergehenden Möglichkeiten bergen ein erhebliches Risiko für Datendiebstahl, unbefugten Zugriff und Cyberangriffe. Obwohl laut einer Umfrage [1, S. 23] die Hälfte der Erwachsenen weltweit glauben, dass die von ihnen ergriffenen Maßnahmen ausreichen, um sich gegen Identitätsdiebstahl zu schützen, sind 63 Prozent darüber besorgt, dass ihre Identität gestohlen wird. Weiter fühlen sich knapp sieben von zehn Menschen heute anfälliger für Identitätsdiebstahl als noch vor ein paar Jahren. Ein wesentlicher Grund für die Zunahme von Identitätsdiebstahl liegt neben zu schwachen Passwörtern hauptsächlich daran, wie Menschen damit umgehen. Bei einer Frage bezüglich der mehrmaligen Verwendung derselben Benutzernamen und Passwörter haben 82 Prozent zugegeben, zumindest manchmal dieselben Anmeldedaten für unterschiedliche Konten zu verwenden. Knapp die Hälfte davon, etwa 45 Prozent, verwenden sogar immer oder in den meisten Fällen dieselben Zugangsdaten [2, S. 12].

Im Rahmen der vorliegenden Studienarbeit wird eine Analyse und Implementierung einer Multi-Faktor-Authentifizierung mit Shamir's Secret Sharing durchgeführt. Dazu wird zu Beginn auf unterschiedliche Authentifizierungsarten eingegangen, einschließlich relevanter Fakten sowie der jeweiligen Vor- und Nachteile. In einem weiteren Schritt erfolgt eine kurze Beschreibung der Methode inklusive einer mathematischen Veranschaulichung, um ein gemeinsames Verständnis für die nachfolgende Analyse der Implementierung zu erlangen. Daran anknüpfend führt eine Betrachtung relevanter Sicherheitsaspekte zu einer Zusammenfassung.

II. Authentifizierung mit Faktoren

A. Ein-Faktor-Authentifizierung

Lange Zeit haben Authentifizierungsmethoden auf einem einzelnen Identifikationsfaktor beruht, in der Regel einer Kombination aus Benutzername und Passwort. Wenn diese beiden Parameter über mehrere Dienste hinweg identisch sind, bedeutet dies, dass ein Angreifer, der ein einziges Konto kompromittiert, automatisch Zugriff auf die anderen Konten erhält — Dabei spielt die Stärke des Passworts keine Rolle. Dieser One-Factor-Authentication (OFA) Ansatz hat jahrzehntelang das Rückgrat der Informationssicherheit gebildet. Dennoch hat sich angesichts der zunehmenden Komplexität von Cyberbedrohungen gezeigt, dass die Abhängigkeit von einem einzigen Faktor für die Authentifizierung eine Schwachstelle darstellt, die anfällig für verschiedene Verletzungen wie Brute-Force-Angriffe, Phishing und Keylogging ist. Diese Schwachstellen verdeutlichen, dass OFA für heutige Anwendungsfälle in aller Regel keine ausreichende Sicherheit mehr bietet.

B. Multi-Faktor-Authentifizierung

Multi-Factor-Authentication (MFA) stellt einen signifikanten Fortschritt in der Evolution der digitalen Sicherheitsmaßnahmen dar. Im Gegensatz zur Einzelfaktor-Authentifizierung, die üblicherweise auf einer einzigen Form des Nachweises wie einem Passwort basiert, erhöht MFA die Sicherheit durch zusätzliche Schutzebenen, indem mehrere unabhängige Zugangsdaten für die Authentifizierung erforderlich sind. Diese Zugangsdaten können in drei Hauptkategorien eingeteilt werden:

1) *Wissen*: Informationen, die der Benutzer kennt, wie Passwörter, PINs und Antworten auf geheime Fragen.

2) *Eigenheit*: Biologische Merkmale, die einzigartig für den Benutzer sind, wie Fingerabdrücke, Netzhautmuster oder Gesichtserkennung.
3) *Besitz*: Gegenstände oder Geräte, die der Benutzer besitzt, wie Smartphones, Chipkarten oder physische Schlüssel. Die Bestätigung des Besitzes kann verschiedene Formen annehmen, angefangen von der Entgegennahme und Eingabe eines per SMS an eine registrierte Telefonnummer gesendeten Codes bis hin zum Einsetzen eines physischen Schlüssels in ein Schloss.

Der Hauptvorteil von MFA gegenüber OFA liegt daher im schichtbasierten Ansatz. Selbst wenn ein Angreifer es schafft, einen Authentifizierungsfaktor zu umgehen, bieten die verbleibenden Faktoren weiterhin Schutz. Eine Kompromittierung eines Faktors gefährdet also nicht die Gesamtsicherheit. Trotz der Stärken bringt eine MFA auch eigene Herausforderungen mit sich, wie beispielsweise die potenziell erhöhte Komplexität und die Notwendigkeit für Benutzer, mehrere Authentifizierungsfaktoren zu verwalten. Dennoch überwiegen die Vorteile der MFA oft diese potenziellen Nachteile, insbesondere in Umgebungen, in denen der Schutz sensibler Daten oberste Priorität hat.

III. Shamir's Secret Sharing

Secret Sharing ist ein grundlegender Baustein der modernen Kryptographie. Eines der bekanntesten Verfahren wurde am 1. November 1979 veröffentlicht und ist nach seinem Erfinder Adi Shamir, einem israelischen Kryptographen, benannt: Shamir's Secret Sharing [3].

Es basiert auf der Idee, ein Geheimnis in mehrere Teile, sogenannte Shares, aufzuteilen. Um das Geheimnis wiederherzustellen, müssen eine bestimmte Anzahl dieser Shares zusammengebracht werden. Jeder einzelne Share ist für sich genommen bedeutungslos und gibt keinerlei Informationen preis. Ein Schwellenwert definiert die minimale Anzahl von Shares, die erforderlich sind, um das Geheimnis wiederherstellen zu können. Dies stellt sicher, dass das Geheimnis selbst dann sicher bleibt, wenn ein Teil der Shares verloren gehen oder in die Hände eines Angreifers gelangen. Das dabei verwendete (k, n)-Schwellenwertschema legt fest, wie viele k Shares benötigt werden, um auf das Geheimnis zu kommen, n ist größer k und bezieht sich auf die Gesamtzahl der Shares, in die das Geheimnis aufgeteilt wird.

A. Mathematische Veranschaulichung

Shamir's Secret Sharing basiert auf dem Prinzip der Polynominterpolation in endlichen Körpern, wobei k Punkte ein Polynom vom Grad $k-1$ eindeutig definieren. Um dies anhand eines mathematischen Beispiels zu veranschaulichen, wird im Folgenden ein (2, 3)-Schwellenschema mit $k = 2$ und $n = 3$ betrachtet, bei dem das Geheimnis S der Zahl 42 entspricht. Sei $p = 43$ eine Primzahl mit $p > S$. Alle Berechnungen erfolgen im endlichen Körper $GF(p)$.

1) Generierung der Shares: Der erste Schritt besteht darin, ein Polynom vom Grad $k - 1 = 2 - 1 = 1$ aufzustellen:

$$f(x) = mx + b \mod p$$

Die Konstante b entspricht dabei dem Geheimnis S. Aus Gründen der Übersichtlichkeit wird in diesem Beispiel $m = 4$ gewählt:

$$f(x) = 4x + 42 \mod 43$$

Im nächsten Schritt erfolgt die Berechnung von n Punkten in der Ebene. Dazu wird für $x = 1...n$ eingesetzt:

$$\text{Für } x = 1 : y_1 = f(1) = 4 * 1 + 42 \mod 43 = 3$$
$$\text{Für } x = 2 : y_2 = f(2) = 4 * 2 + 42 \mod 43 = 7$$
$$\text{Für } x = 3 : y_3 = f(3) = 4 * 3 + 42 \mod 43 = 11$$

Jeder der berechneten Punkte $(1, 3), (2, 7), (3, 11)$ repräsentiert dabei einen Share.

2) Rekonstruktion mit linearem Gleichungssystem: Sind nun k Punkte gegeben, kann das ursprüngliche Geheimnis rekonstruiert werden. Im Folgenden werden die Punkte $(1, 3)$ und $(2, 7)$ als Gleichungen in einem linearen Gleichungssystem dargestellt:

$$1.\ m + b = 3$$
$$2.\ 2m + b = 7$$

Die Unbekannten werden nun über das Substitutionsverfahren gelöst. Durch Umstellen der ersten Gleichung nach b folgt $b = 3 - m$. Dieser Ausdruck wird in die zweite Gleichung eingesetzt, was zu $2m + (3 - m) = 7$ führt. Daraus folgt $m = 4$. Die erhaltene Lösung für m wird dann in die umgestellte erste Gleichung eingesetzt, um b zu berechnen: $b = 3 - 4 = -1$. Da die Berechnungen im endlichen Körper $GF(43)$ durchgeführt werden, wird das Ergebnis modulo 43 genommen, um das Geheimnis im Wertebereich von 0 bis $p - 1$ zu erhalten: $b = -1 \mod 43 = 42$, was dem Geheimnis $S = 42$ entspricht. Bei größeren Werten von k würde ein Polynom höheren Grades und ein entsprechend größeres lineares Gleichungssystem entstehen.

3) Rekonstruktion mit Lagrange-Interpolation: Die Lagrange-Interpolation ist das in der Praxis am häufigsten eingesetzte Verfahren zur Bestimmung des Polynoms einer bestimmten Ordnung, das durch eine gegebene Menge von Punkten verläuft. Diese Methode bietet den Vorteil, dass sie direkt eine Formel zur Rekonstruktion des Geheimnisses liefert, ohne dass das Gleichungssystem explizit gelöst werden muss. Beide Methoden führen jedoch zum gleichen Ergebnis.

IV. Die Idee der Kombination

Das Ziel dieser Arbeit besteht darin, die Vorteile beider Verfahren zu kombinieren, um eine robuste und sichere Authentifizierungsmethode zu entwickeln.

Abbildung 1 zeigt den groben Ablauf, wie die Shares erzeugt werden. Im Kern wird ein Geheimnis aus mehreren Authentifizierungsfaktoren generiert und mithilfe von

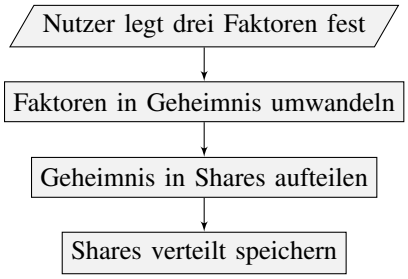

Abbildung 1. Überblick: Generierung der Shares

Shamir's Secret Sharing aufgeteilt. Die zur Multi-Faktor-Authentifizierung verwendeten Faktoren entsprechen einem Passwort (Wissen), dem Fingerabdruck des Nutzers (Eigenheit) und einem Wiederherstellungsschlüssel in Form eines QR-Codes (Besitz). Die Kombination aller drei Faktoren dient als Grundlage zur Berechnung des Geheimnisses, welches anschließend mittels Shamir's Secret Sharing in drei Shares aufgeteilt wird, wobei nur zwei Stück zur späteren Authentifizierung benötigt werden. Jeder Share wird anschließend eindeutig zu einem der obigen Faktoren zugeordnet und auf unterschiedlichen Wegen sicher abgelegt, zum Beispiel der Passwort-Share in einer Datenbank, der Fingerabdruck-Share im sicheren Bereich eines Smartphones und der zugehörige Share zum Wiederherstellungsschlüssel als QR-Code ausgedruckt an einem sicheren Ort.

Nachdem alle Shares erfolgreich generiert wurden und gespeichert sind, kann sich der Nutzer mit zwei der drei zu Beginn festgelegten Faktoren authentifizieren. Mit Blick auf Abbildung 2 wählt der Nutzer zu Beginnn aus, welche zwei Faktoren er dazu verwenden möchte. Nach Eingabe eines korrekten Faktores gibt das System den verknüpften Share frei, welcher im Anschluss zur Rekonstruktion verwendet wird. Sobald beide Shares zur Verfügung stehen wird das Geheimnis wiederhergestellt und überprüft, ob der Wert dem ursprünglichen Geheimnis entspricht. Falls ja, ist die Authentifizierung erfolgreich, andernfalls erhält der Nutzer eine Fehlermeldung.

A. Vorteile

1) Erhöhte Sicherheit: MFA und SSS ergänzen sich gegenseitig, um eine robuste Sicherheitsarchitektur zu schaffen. Während MFA bereits eine zusätzliche Sicherheitsebene durch die Verwendung mehrerer Faktoren bietet, stellt Shamir's Secret Sharing sicher, dass die zu schützenden Daten unverschlüsselt dezentral gespeichert werden können, da ein einzelner Share keine Rückschlüsse auf das Geheimnis zulässt. Dadurch wird das Risiko eines vollständigen Datenlecks oder unbefugten Zugriffs drastisch minimiert.

2) Flexibilität: Benutzer können aus drei verschiedenen Optionen (Passwort, Fingerabdruck und Wiederherstellungsschlüssel) zur Authentifizierung wählen. Darüber hinaus kann die Aufteilung von Shares an unterschiedliche Geräte oder Personen erfolgen, um sowohl den Bedürfnissen und Anforderungen des Nutzers als auch eines Systems gerecht zu werden.

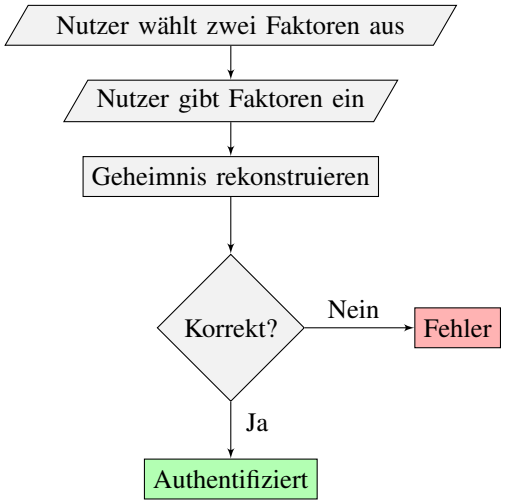

Abbildung 2. Überblick: Authentifizierung mit Shares

3) Schutz vor Datenverlust: Wenn beispielsweise ein Benutzer das verknüpfte Smartphone verliert oder es irreparabel beschädigt wurde, ist weiterhin ein Zugriff über die beiden anderen Faktoren gewährleistet. Dies bietet einen zusätzlichen Schutz vor Datenverlust.

Zusammenfassend lässt sich sagen, dass das Zusammenspiel beider Methoden die Sicherheit eines Authentifizierungsprozesses erhöht. Deshalb wird im Folgenden eine mögliche Implementierung vorgestellt, welche die einzelnen Schritte im Detail verdeutlichen soll.

V. Implementierung

In diesem Kapitel wird die Idee der Kombination von MFA und SSS anhand einer Implementierung in Python v3.10.9 schrittweise erklärt. Die folgenden Code-Ausschnitte dienen nur zur Veranschaulichung und sind deshalb ohne weitere Vorkehrungen nicht zwingend lauffähig.

A. Phase 1: Konstruktion des Geheimnisses

Bevor eine Authentifizierung stattfinden kann, müssen die dafür benötigten Shares erst einmal erzeugt werden. Dazu wird ein Geheimnis benötigt, welches auf allen drei Faktoren basiert. Das Ziel dieser Phase ist es, eine natürliche Zahl zu konstruieren, die als Geheimnis für Shamir's Secret Sharing verwendet werden kann.

1) Nutzer legt drei Faktoren fest: Zu den eben genannten Faktoren zählen ein Passwort, ein Fingerabdruck und ein Wiederherstellungsschlüssel. Wie in Listing 1 abgebildet müssen die ersten beiden Faktoren vom Nutzer eingegeben werden, der Dritte wird zufällig in Form eines 16 Byte langen hexadezimalen Strings generiert.

```
password = input() # 1. Faktor                          1
fingerprint = input() # 2. Faktor                       2
recovery_key = os.urandom(16).hex() # 3. Faktor         3
```

Listing 1. Initialisierung der drei Faktoren

2) Faktoren umwandeln: All diese Faktoren werden im späteren Verlauf für die Authentifizierung benötigt. Daher ist es wichtig, dass diese Informationen umgewandelt werden, um mögliche Rückschlüsse auf die ursprünglichen Eingaben des Nutzers auszuschließen. Aus diesem Grund werden im nächsten Schritt alle Faktoren mit SHA256 gehasht (siehe Abbildung 3).

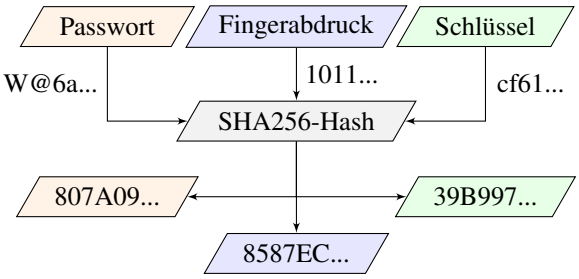

Abbildung 3. Faktoren umwandeln

Die in Listing 2 gegebene Funktion `hash_string` nimmt einen String `value` entgegen. Die Funktion greift auf die `hashlib`-Bibliothek zurück, um den SHA-256-Hash des gegebenen Werts zu berechnen. Zunächst wird der übergebene Wert mit `.encode()` in eine Bytefolge umgewandelt, woraufhin die `.digest()`-Methode auf den berechneten Hash angewendet wird, um das Ergebnis als Bytes zurückzugeben, um diese im nachfolgenden Schritt in eine ganze Zahl umwandeln zu können.

```
def hash_string(value):
    return hashlib.sha256(value.encode()).digest()

password_hash = hash_string(password)
fingerprint_hash = hash_string(fingerprint)
recovery_key_hash = hash_string(recovery_key)
```

Listing 2. Hashen der drei Faktoren

3) Hashwerte in Zahlen umwandeln: Alle drei erhaltenen Hashwerte müssen nun als Zahlen interpretiert werden, da Shamir's Secret Sharing eine ganze Zahl für das Geheimnis fordert. Die Funktion `hash_to_int` aus Listing 3 nimmt ebenfalls einen Wert `value` entgegen, der hier allerdings die zuvor generierte Bytefolge repräsentiert. Durch Verwendung der Methode `int.from_bytes()` mit dem Parameter `value` wandelt die Funktion diese Bytefolge in eine Ganzzahl um. Dabei erfolgt die Interpretation der Bytes in der Reihenfolge „big", wodurch das Most Significant Bit zuerst und das Least Significant Bit zuletzt berücksichtigt wird. Das Ergebnis, also die umgewandelte Ganzzahl, wird von der Funktion zurückgegeben.

```
def hash_to_int(value):
    return int.from_bytes(value, byteorder="big")

password_number = hash_to_int(password_hash)
fingerprint_number = hash_to_int(fingerprint_hash)
recovery_key_number = hash_to_int(recovery_key_hash)
```

Listing 3. Hashwerte als Zahlen interpretieren

4) Geheimnis erzeugen: Der vorletzte Schritt besteht darin, alle Zahlen zusammenzufügen, um das Geheimnis zu erhalten. Dabei ist zu beachten, dass die Zahlen nicht addiert, sondern in zufälliger Reihenfolge konkateniert werden. Dazu wird eine Liste mit den Zahlen der drei umgewandelten Hashwerte erstellt und anschließend gemischt. Die nun zufällig angeordneten Zahlen werden in einer Schleife durchlaufen, in einen String umgewandelt und aneinandergereiht. Dieser String wird zum Schluss wieder zu einem Integer konvertiert und der Variable für das Geheimnis zugewiesen (siehe Listing 4).

```
numbers = [password_number, fingerprint_number,
    recovery_key_number]
random.shuffle(numbers)
S = int("".join(str(num) for num in numbers))
```

Listing 4. Zahlen zu Geheimnis konkatenieren

Um das Geheimnis während der Authentifizierung bei einer erfolgreichen Rekonstruktion auf Korrektheit prüfen zu können, muss es gespeichert werden. Dazu wird es wie die Faktoren in Listing 2 mit SHA256 gehasht.

B. Phase 2: Generierung der Shares

Das nicht gehashte Geheimnis wird in dieser Phase dazu benötigt, um es unter Verwendung von Shamir's Secret Sharing in einzelne Shares aufteilen zu können.

1) Primzahl erzeugen: Alle Berechnungen erfolgen wie auch zu Beginn in der mathematischen Veranschaulichung in einem endlichen Körper. Dieser wird definiert als $GF(p)$, wobei p eine Primzahl in derselben Größenordnung des Geheimnisses ist. Die Bibliothek „sympy" bietet dafür die Funktion „nextprime". Die Eingabe der Variable S erzeugt die nächstgrößere Primzahl, welche für die nachfolgenden Berechnungen verwendet wird.

2) Shares erzeugen: Nach allen Vorbereitungen wird das Geheimnis nun im letzten Schritt in einzelne Shares aufgeteilt. Das in Listing 5 verwendete (2, 3)-Schwellenwertschema erzeugt insgesamt drei Shares, wovon zwei zur Rekonstruktion benötigt werden.

```
def create_shares(S, p):
    m = int.from_bytes(os.urandom(32),
        byteorder="big") # Zufälliger Koeffizient
    shares = []
    for x in range(1, 4): # x aufsteigend iterieren
        y = (m * x + S) % p # y-Wert berechnen
        shares.append((x, y)) # Punkt hinzufügen
    return shares

shares = create_shares(S, p)
```

Listing 5. Shares erzeugen

LITERATUR

[1] Gen Digital Inc., „2023 Norton Cyber Safety Insights Report," Feb. 2023. Adresse: https://filecache.mediaroom.com/mr5mr_nortonlifelock/178041/2023%20NCSIR%20US-Global%20Report_FINAL.pdf (besucht am 05. 06. 2023).

[2] Morning Consult und IBM Security, „Security Side Effects of the Pandamic,“ Juli 2021. Adresse: https://filecache.mediaroom.com/mr5mr_nortonlifelock/178041/2023%20NCSIR%20US-Global%20Report_FINAL.pdf (besucht am 05. 06. 2023).

[3] A. Shamir, „How to share a secret,“ *Communications of the ACM*, Jg. 22, Nr. 11, S. 612–613, 1. Nov. 1979, ISSN: 0001-0782. DOI: 10.1145/359168.359176. Adresse: https://dl.acm.org/doi/10.1145/359168.359176 (besucht am 20. 06. 2023).