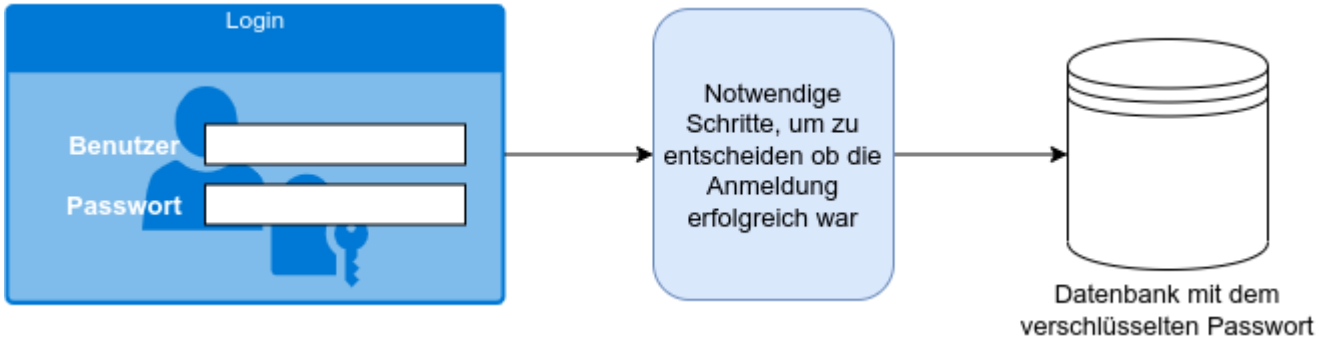
Passwörter speichern

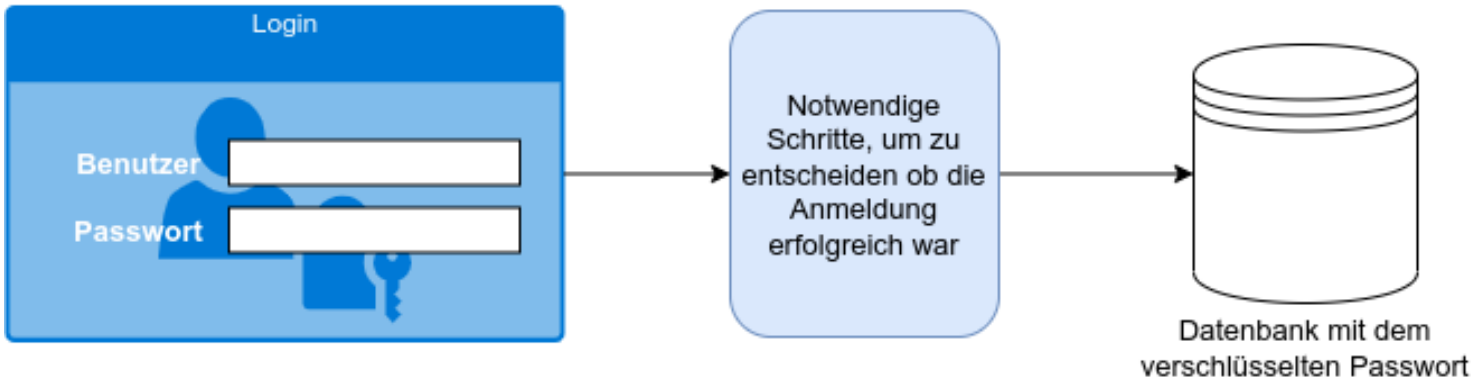
Für deine Microblogging Plattform möchtest du die Zugangsdaten deiner Nutzer in der einer Datenbank speichern, insbesondere den Benutzernamen und das Passwort, das die Benutzer zur Anmeldung verwenden. weil du in Informatik gut aufgepasst hast, ist dir sofort klar, dass es nicht in Frage kommt, die Passwörter unverschlüsselt in der Datenbank abzulegen.

Dein erster Gedanke ist: Die speichere ich **verschlüsselt** in der Datenbank ab! Die Situation stellt sich also wie folgt dar:



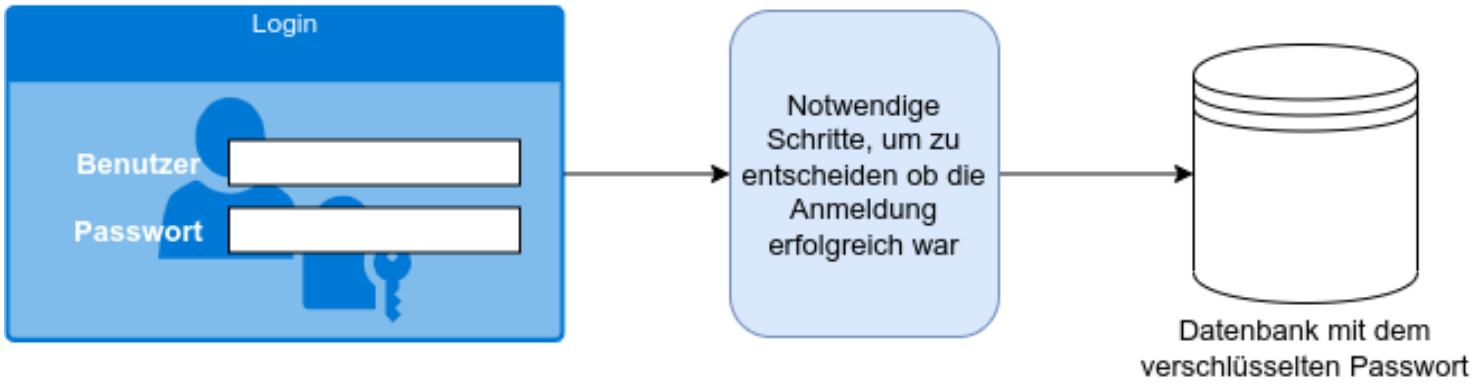
Passwörter speichern

Dein erster Gedanke ist: Die speichere ich **verschlüsselt** in der Datenbank ab!
Die Situation stellt sich also wie folgt dar:



Passwörter speichern

(A1)



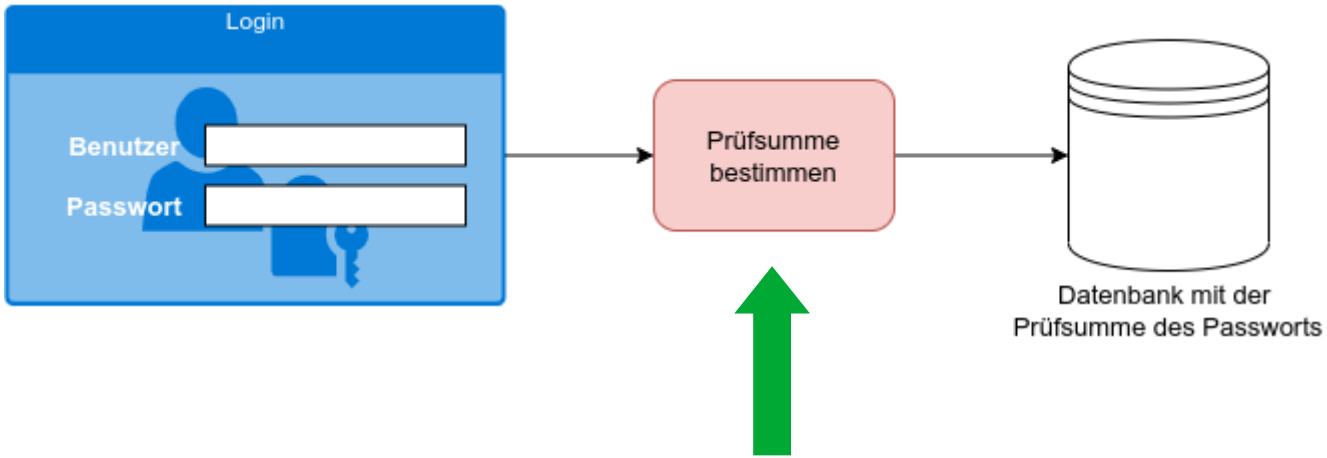
- Überlege dir, wie der Vorgang ablaufen könnte, der dazu führt, den Anmeldeversuch bei Eingabe des richtigen Passworts als „korrekt“ zu bewerten.
- Welche Probleme erkennst du, wenn das Passwort verschlüsselt in der Datenbank gespeichert wird?

https://wiki.qg-moessingen.de/hashes

Die „Prüfsummenstrategie“

Passwörter verschlüsselt zu speichern macht keinen Sinn – man benötigt eine andere Methode.

Die „Prüfsummenstrategie“:



Was nimmt man da geschickt als „Prüfsumme“?

Die „Prüfsummenstrategie“

Dein Entwicklungschef schlägt dir zwei Möglichkeiten vor:

- Die **iterierte Quersumme** ist die Quersumme, die entsteht, wenn man solange immer wieder die Quersumme ausrechnet, bis nur noch eine einzige Ziffer übrig bleibt. Beispiel: Für die Zahl 97 lautet die normale Quersumme 16, berechnet man davon wiederum die Quersumme, so entsteht die iterierte Quersumme: 7.
- Die **alternierende Quersumme** entsteht durch abwechselndes Subtrahieren und Addieren der einzelnen Ziffern. Beispiel: Für die Zahl 1234 ist die alternierende Quersumme 1 - 2 + 3 - 4 = -2).

Um unsere Überlegungen einfach zu halten, lassen wir fürs Erste nur Kennwörter zu, die aus Zahlen bestehen.

Die „Prüfsummenstrategie“

Benutzername	Passwort	Iterierte QS	Alternierende QS
martin	12345	1+2+3+4+5 = 15 -> 1+5 = **6**	1-2+3-4+5 = **3**
susi	123456		
franzi	12223345678		
karle	123456789		
eva	1234567890		
kathrin	11111121		
hubertus	191817		

(A2)

- Ergänze die Tabelle für die weiteren Zeilen
- Entscheide für welche Methode der Passwortspeicherung du dich entscheiden würdest. Begründe deine Entscheidung.
- Welche Eigenschaft würdest du dir von einer optimalen Prüfsumme wünschen? Erläutere, warum diese auf deiner Wunschliste stehen würde

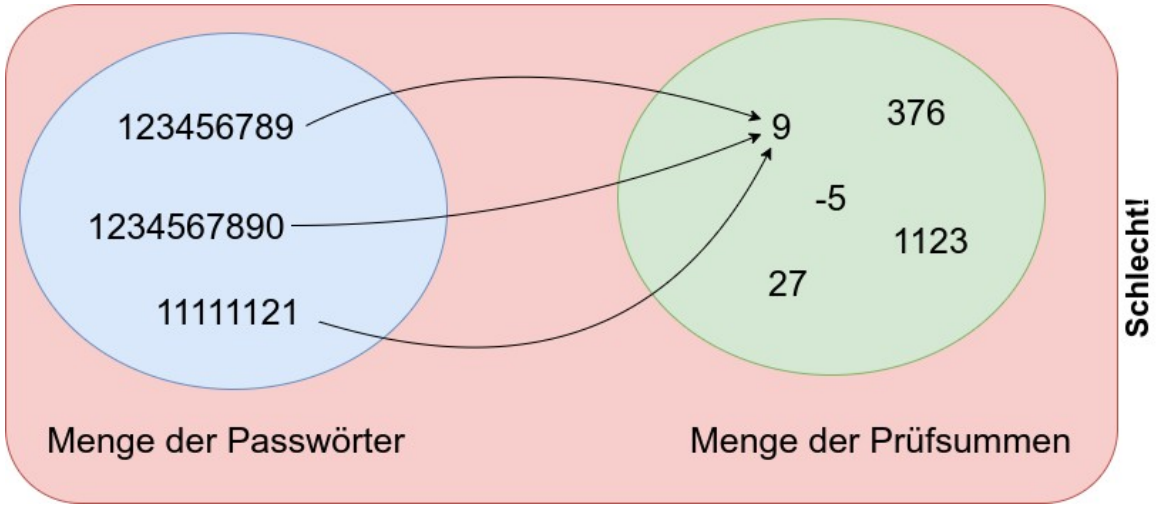
Die „Prüfsummenstrategie“

Benutzername	Passwort	Iterierte QS	Alternierende QS
martin	12345	1+2+3+4+5 = 15 -> 1+5 = **6**	1-2+3-4+5 = **3**
susi	123456	3	-3
franzi	12223345678	7	5
karle	123456789	9	5
eva	1234567890	9	5
kathrin	11111121	9	1
hubertus	191817	9	-21

https://wiki.qg-moessingen.de/hashes

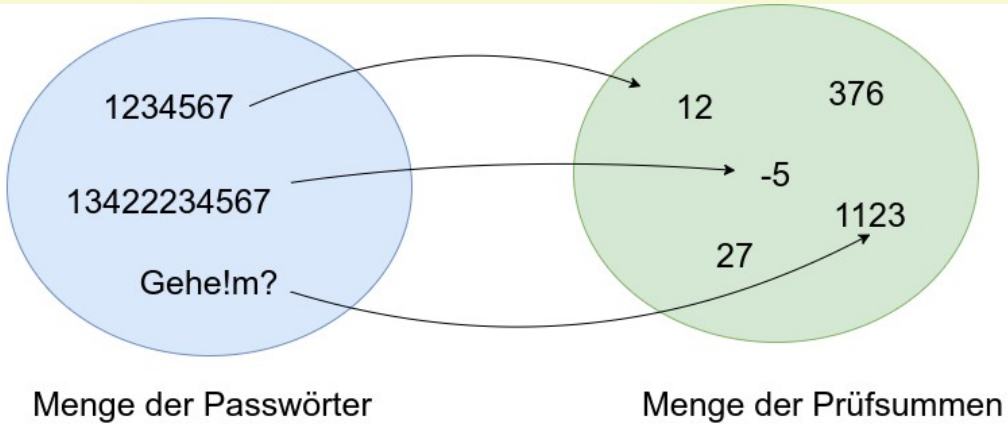
Die „Prüfsummenstrategie“

Wunsch: Jedes Passwort soll eine andere Prüfsumme haben



Fachsprache: Wenn zwei Eingaben (Passwörter) dieselbe Prüfsumme haben spricht man von einer **Kollision-**

Wunsch: Unsere Prüfsumme soll kollisionsfrei sein.



Die „Prüfsummenstrategie“

Deine Entwickler stimmen dir zu, dass das so keinen Sinn macht und schlagen weitere Prüfsummen vor, die weniger anfällig für Kollisionen sein sollen:

Benutzername	Passwort	Methode 1	Methode 2	Methode 3
martin	12345	54321	34567	120
susi	123456	654321	345678	720
franzi	1222334567	7654332221	3444556789	60480
karle	4534567	7654354	6756789	50400
eva	657703	307756	879923	4410
kathrin	11111121	12111111	33333343	2
hubertus	191817	718191	31131039	504

(A3)

- Finde heraus, welche Methoden zur „Prüfsummenbildung“ hier zum Einsatz kommen.
- Wenn du dich für eine dieser drei Methoden entscheiden müsstest - welche wäre dies? Begründe deine Wahl.

Die „Prüfsummenstrategie“

Wunsch 1: Unsere Prüfsumme soll kollisionsfrei sein.

Neu: Idealerweise sollte man aus der Prüfsumme nicht auf das ursprüngliche Passwort schließen können. Bei den beiden ersten Methoden ist das der Fall, beim Querprodukt nicht.

Daraus ergibt sich ein weiterer Wunsch an unsere „Prüfsumme“

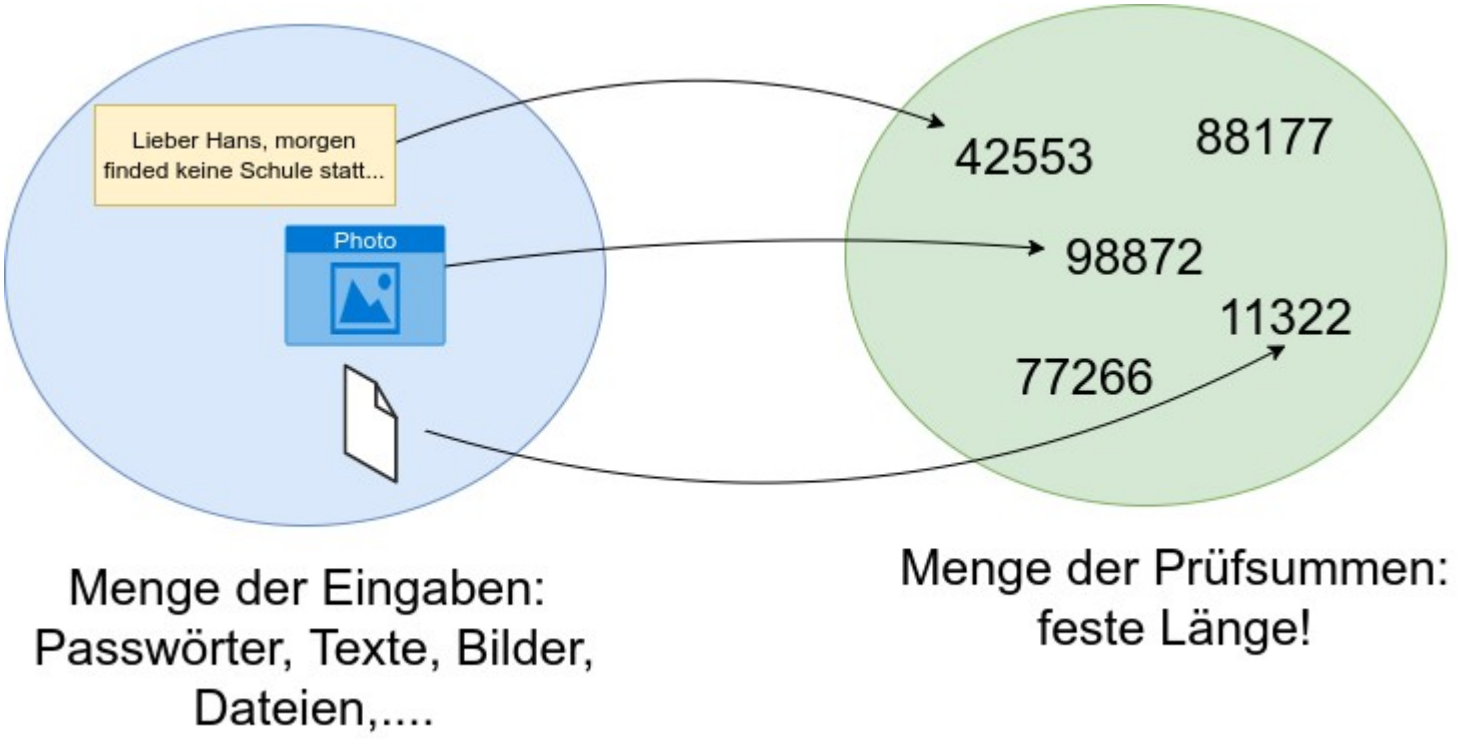
Wunsch 2: Unsere Prüfsumme soll unumkehrbar sein.

Hashfunktionen

In der Kryptographie spielen sogenannte **kryptographische Hashfunktionen** eine große Rolle (wie wir bei genauerer Betrachtung noch sehen werden…)

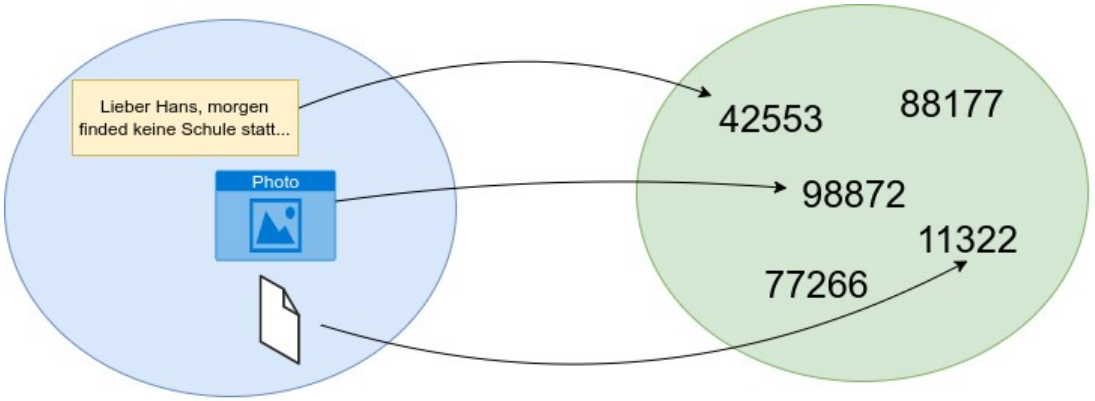
Diese Hashfunktionen sind unseren „Prüfsummen“ nicht unähnlich, haben aber noch eine erschwerende **Randbedingung** zu erfüllen:

Es sollen beliebig große Eingaben auf eine feste Zahl von Zeichen abgebildet werden:



Hashfunktionen

Es sollen beliebig große Eingaben auf eine feste Zahl von Zeichen abgebildet werden!



Wunsch 1: Unsere Prüfsumme soll kollisionsfrei sein.

Wunsch 2: Unsere Prüfsumme soll unumkehrbar sein.

(A4)

Überlege dir, welche Auswirkungen diese „Randbedingung“ auf unsere beiden Wünsche an die „Prüfsummenfunktion“ haben.

Hashfunktionen

Kryptographische Hashfunktionen:

- … bilden eine **beliebig große Eingabemenge** auf **Hashes fester Länge** ab. Meist erfolgt die Ausgabe Hexadezimal.
- … sind **nicht umkehrbar**
- … sind **kollisionsarm**. Insbesondere soll es **unmöglich** sein **Kollisionen zu konstruieren**, also Eingaben finden zu können, die denselben Hash Wert haben

Beispiele:

- MD5 (gebrochen – man kann Kollisionen konstruieren)
- SHA1 (unsicher, kommt derzeit bei GIT zum Einsatz)
- SHA2 (mehrere Varanten, gilt als weitgehend sicher)
- Bcrypt (wird zum Passworthashing verwendet)

Hashfunktionen

(A5)

Beschreibe stichwortartig, was bei der Registrierung eines neuen Benutzers an deinem sozialen Netzwerk geschehen muss, um den neuen Benutzer in der Datenbank einzutragen. Was gibt der Benutzer ein, was macht das Softwaresystem, was wird in die Datenbank eingetragen?

Verfahre ebenso beim Anmeldevorgang eine bestehenden Benutzers: Was gibt der Benutzer ein, was macht das Softwaresystem, was wird aus der Datenbank gelesen?