

Erfahrung & Expertise: Ein Blick auf meinen Werdegang



Dr. Thorsten Weber



Beruflicher Werdegang:

- **Team Lead** Offensive Security & Awareness, SAP SE, Walldorf
- **Team Lead** IT-Security, oculavis GmbH, Aachen
- **IT-Security** Administrator, NATO, Brüssel



Akademischer Werdegang:

- **Promotion** an der Universidad Católica San Antonio de Murcia (UCAM) in Kooperation mit der FOM
- **B.Sc. / M.Sc.**, RWTH Aachen Informatik



Nebenberufliche Tätigkeit

- **Dozent**, FOM Hochschule, Aachen
- **ISO/IEC 27001 Lead Auditor**

Integritätsprüfung in verteilten Systemen mit Merkle Trees

Dr. Thorsten Weber
Juli 16, 2025



Was sollten Sie nach dieser Vorlesung wissen und können?



Strukturverständnis

Die Studierenden können den **Aufbau und die Funktionsweise** eines Merkle Trees **beschreiben und erklären**.



Anwendungskompetenz

Die Studierenden können **Anwendungsfälle** von Merkle Trees **identifizieren und deren Nutzen zur Integritätsprüfung begründet** darstellen.



Agenda

1

Einleitung: Wir organisieren ein Konzert

2

Merkle Trees - Grundlagen

3

Merkle Trees – Praxiseinsatz

4

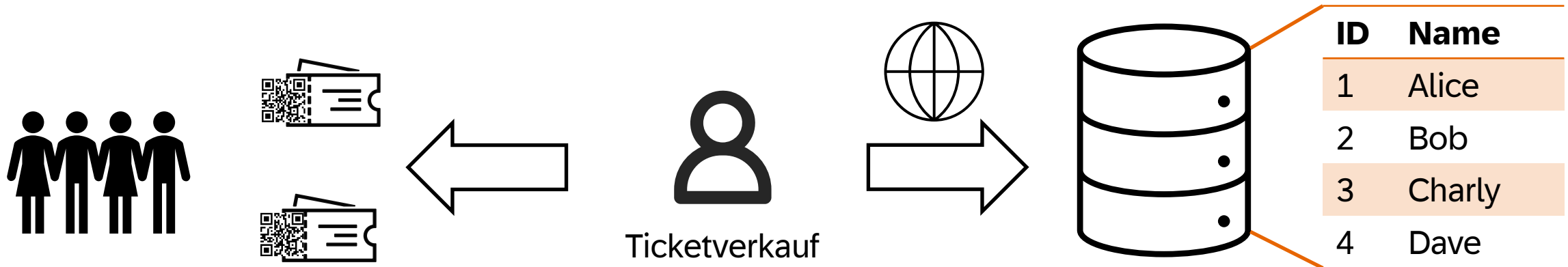
Zusammenfassung, Ausblick und Live Demo

1

Einleitung: Wir organisieren ein Konzert

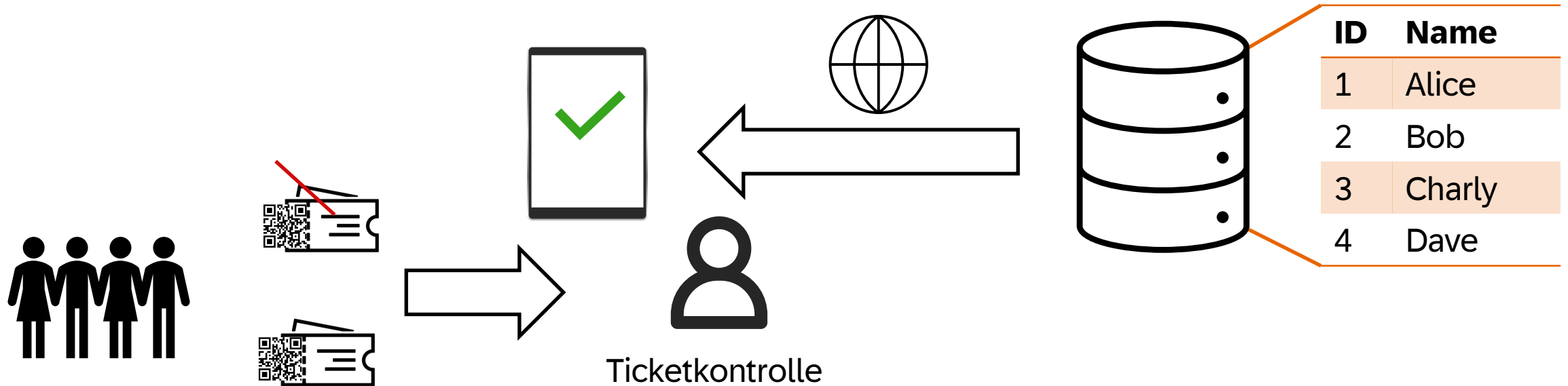
Der Sommer ist da, wir möchten ein Konzert organisieren! - Phase 1

Ein Konzert – 1 Ticketstelle, 1 Einlass, 200 Tickets, Internet vorhanden



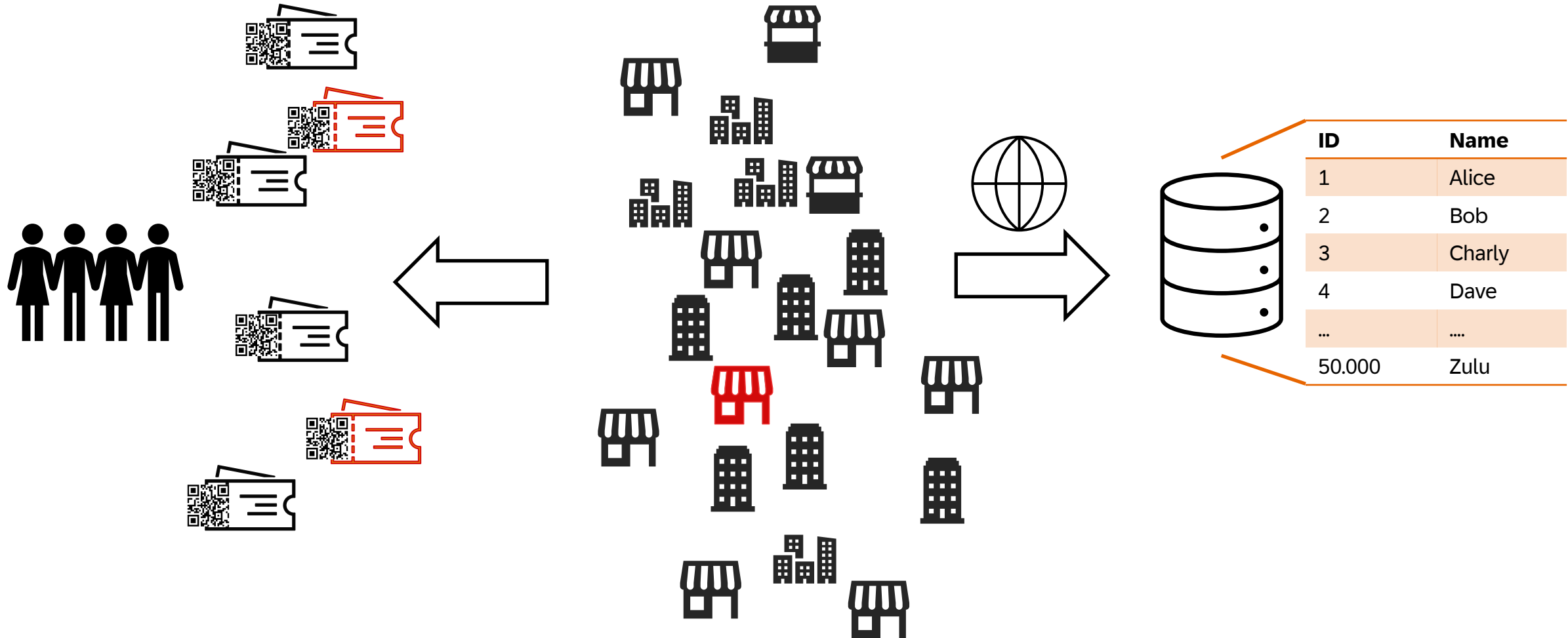
Der Sommer ist da, wir möchten ein Konzert organisieren! - Phase 1

Ein Konzert – 1 Ticketstelle, 1 Einlass, 200 Tickets, Internet vorhanden



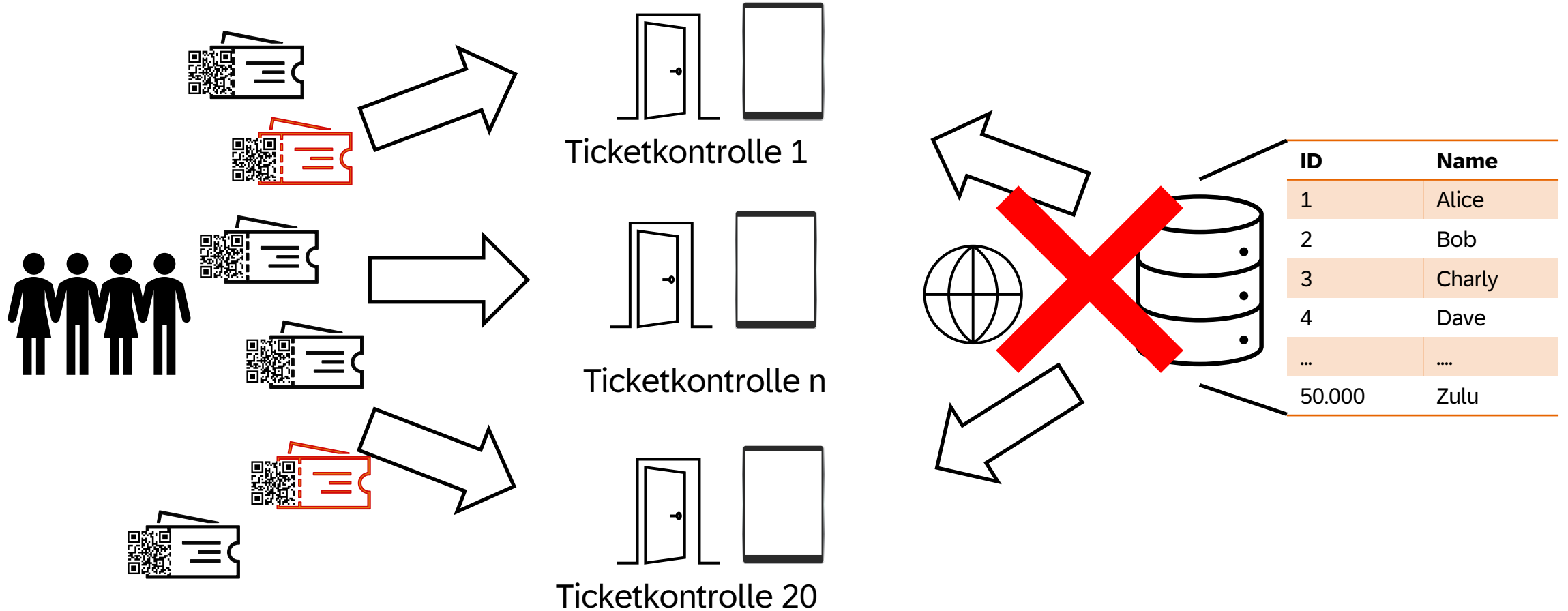
Das Konzert war ein MEGA Erfolg! - Phase 2

Ein Konzert – 200 Ticketstellen, 20 Einlässe, 50.000 Tickets, kein Internet



Das Konzert war ein MEGA Erfolg! - Phase 2

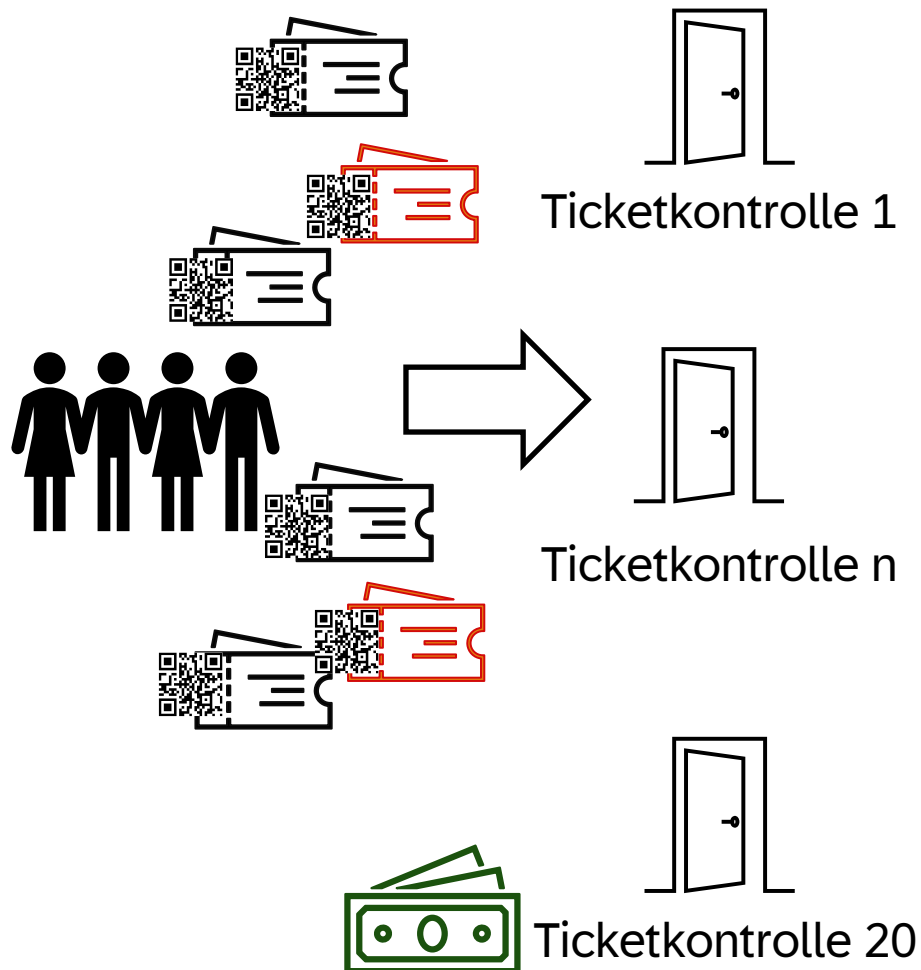
Ein Konzert – 200 Ticketstellen, 20 Einlässe, 50.000 Tickets, kein Internet



Naive Idee: Wir laden die **50.000 Namen** auf alle Ticketscanner (z. B. als CSV oder JSON)

Das Konzert war ein MEGA Erfolg! - Phase 2

Ein Konzert – 200 Ticketstellen, 20 Einlässe, 50.000 Tickets, kein Internet



✓	
✗	
ID	Name
1	Alice
2	Bob
3	Charly
4	Dave
...	...
50.000	Zulu

✓	
✗	
ID	Name
1	Alice
2	Bob
3	Charly
4	Dave
...	...
50.000	Zulu

✓	
✗	
ID	Name
1	Alice
2	Bob
3	Charly
4	Dave
...	...
50.000	Zulu
50.001	Thersten

Frage: Ist das eine gute Idee?

- **Herausforderung 1:** Liste ist groß
⇒ Datenschutz
- **Herausforderung 2:** Was, wenn jemand die Liste nachträglich manipuliert? ⇒ Wie weiß der Scanner, ob er die richtige, originale Liste hat (verteilte, dezentrale Systeme)?

Wie können wir unser Konzert hier absichern?

Ziel ist ein Ansatz, der folgende Anforderungen erfüllt:

- **Speichereffizienz:** Die Liste enthält rund 50.000 Einträge.
- **Verteilbarkeit:** Die Liste muss auf allen Geräten identisch und aktuell gehalten werden.
- **Offline-Fähigkeit und Dezentralität:** Ein Abgleich über das Internet ist nicht möglich.
- **Manipulationssicherheit:**
 - Es dürfen keine Tickets unbemerkt hinzugefügt werden.
 - Kein Eintrag darf gelöscht oder verändert werden.
- **Datenschutzkonformität:** Es muss sichergestellt werden, dass kein Risiko für DSGVO-Verstöße besteht.



2

Merkle Trees - Grundlagen

Ralph Merkle - Pionier der asymmetrischen Kryptographie

- Veröffentlichte **Merkle Trees 1979** in seiner Dissertation [1]
 - **Merkle Tree** – Struktur zur effizienten Integritätsprüfung

United States Patent [19]		[11]	4,309,569
Merkle		[45]	Jan. 5, 1982
Best Available Copy			
[54] METHOD OF PROVIDING DIGITAL SIGNATURES		[56] References Cited	
		U.S. PATENT DOCUMENTS	
[75] Inventor:	Ralph C. Merkle, Mountain View, Calif.	4,200,770 4/1980 Hellman et al. 375/2	
[73] Assignee:	The Board of Trustees of the Leland Stanford Junior University, Stanford, Calif.	<i>Primary Examiner</i> —Howard A. Birmiel <i>Attorney, Agent, or Firm</i> —Flehr, Hohbach, Test, Albritton & Herbert	
[21] Appl. No.:	72,363	[57] ABSTRACT	
[22] Filed:	Sep. 5, 1979	The invention comprises a method of providing a digital signature for purposes of authentication of a message, which utilizes an authentication tree function of a one-way function of a secret number.	
[51] Int. Cl. ³	H04L 9/00		
[52] U.S. Cl.	178/22.08; 178/22.10; 340/825.34		
[58] Field of Search	178/22; 340/149 R, 149 A, 340/152 R; 235/379, 380, 382; 375/2	4 Claims, 1 Drawing Figure	



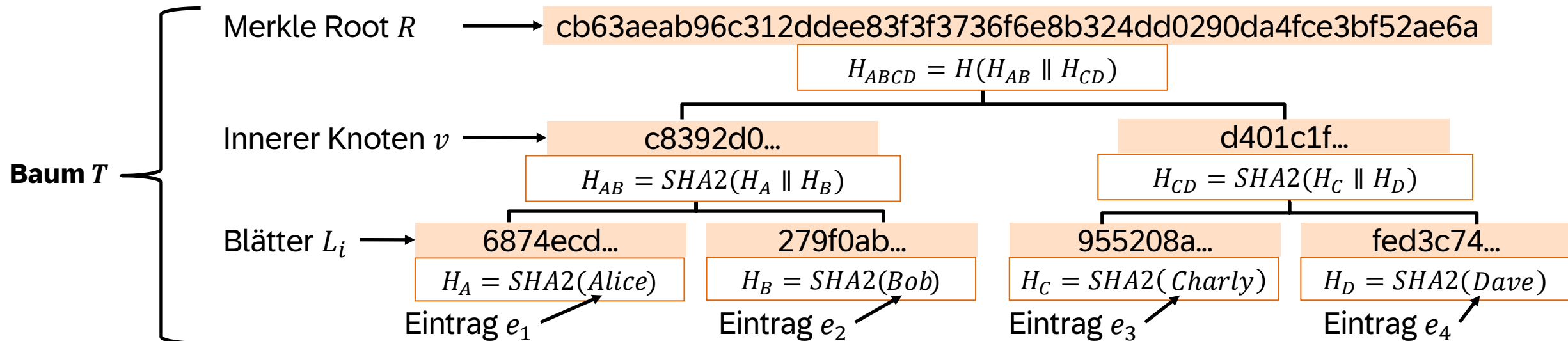
Bildquelle: david.orban, CC BY 2.0, via Wikimedia Commons

Merkle Tree Beispiel

Definition: Merkle Tree [1]

Ein **Merkle-Tree** über einer endlichen Liste von **Einträgen** $E = (e_1, e_2, \dots, e_n)$ ist ein vollständiger binärer **Baum** T , bei dem gilt:

- Die **Blätter** L_i entsprechen den **Hashes** der Einträge: $L_i = H(e_i)$, $i \in \{1, \dots, n\}$
- Jeder **innere Knoten** v mit **linken Kind** v_l und **rechtem Kind** v_r enthält: $v = H(v_l \parallel v_r)$ wobei H eine kryptografische Hashfunktion ist und \parallel die Byteweise-Konkatenation bezeichnet.
- Die **Wurzel** R des Baums heißt **Merkle Root**.

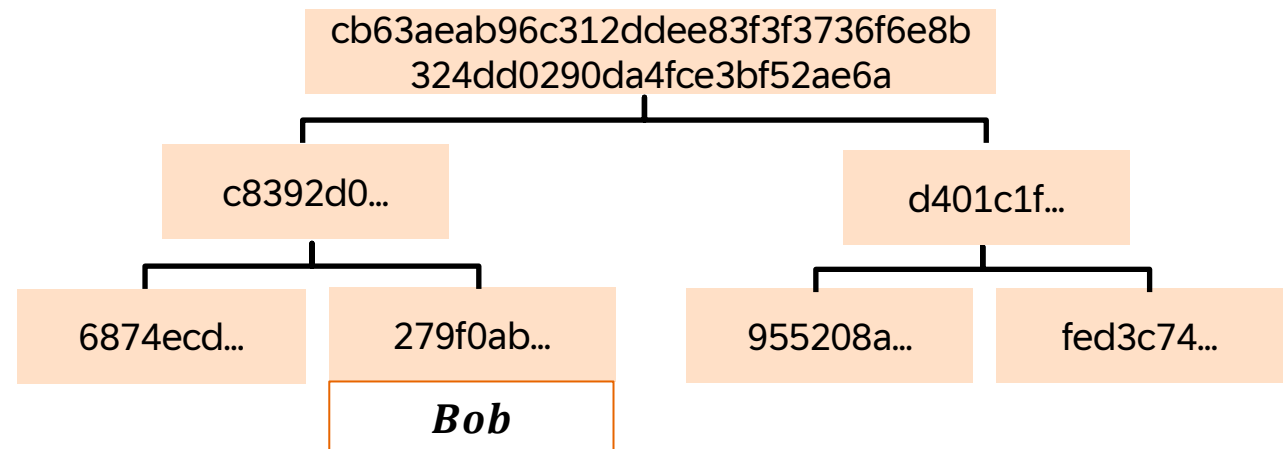


Merkle Proof mit Beispiel für Bob

Definition: Merkle Proof [1]

- Ein Merkle Proof für einen Eintrag e_i ist **eine Folge von Hashes**, die beweist, dass e_i Teil eines Merkle Trees mit der Wurzel **R** ist – ohne dass der gesamte Baum bekannt sein muss.
- Aufbau **Merkle Proof** für den Eintrag e_i :
 - Eintrag e_i ,
 - einer Liste $P = [p_1, \dots, p_k]$ von Hashwerten der Geschwisterknoten auf dem Pfad zur Wurzel,
 - einer Liste $D = [d_1, \dots, d_k]$ mit Richtungshinweisen:
 - $d_j = 0$: der Geschwister-Hash p_j steht **links**,
 - $d_j = 1$: der Geschwister-Hash p_j steht **rechts**.

- $e_i = \text{Bob}$
- $p_1 = 6874ecd\dots, d_0 = 0$
- $p_2 = d401c1f\dots, d_1 = 1$
- $P = [6874ecd\dots, d401c1f\dots]$
- $D = [0, 1]$



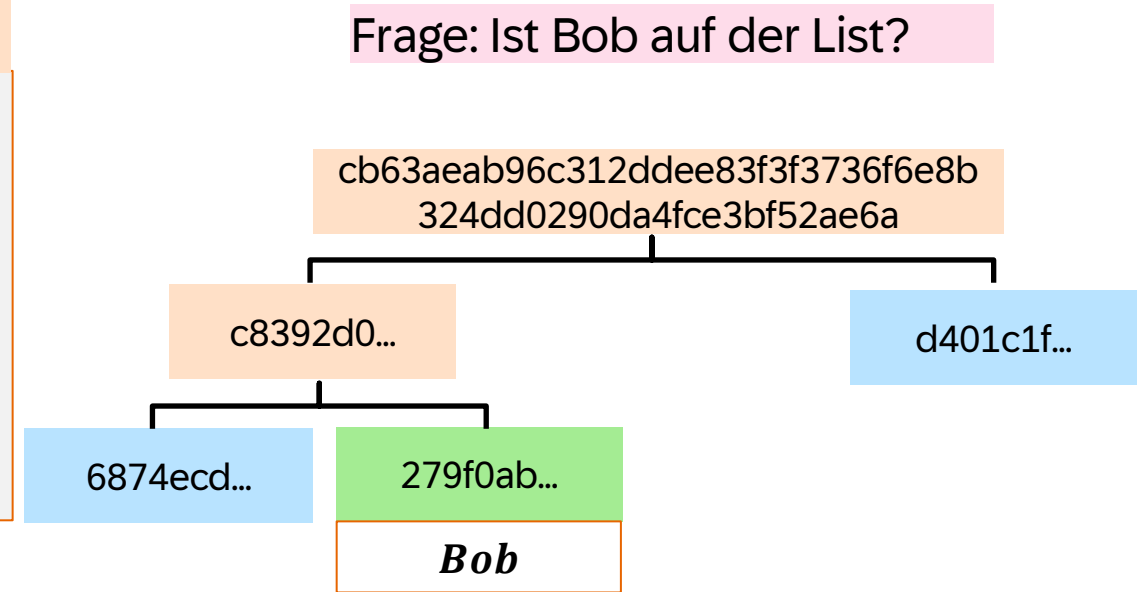
Beispiel Merkle Proof für Bob

Definition: Merkle Proof [1]

Verifikation:

- $v_0 = H(e_i), v_j = \begin{cases} H(p_j \parallel v_{j-1}) & \text{falls } d_j = 0 \\ H(v_{j-1} \parallel p_j) & \text{falls } d_j = 1 \end{cases}$
- Der Beweis gilt als korrekt, wenn: $v_k = R$

Gegeben: $e_i = \text{Bob}$, $P = [6874ecd..., d401c1f...]$, $D = [0, 1]$,
 $R = cb63aeab...$



- $v_0 = 279f0ab...$
- $v_1 = H(p_0 \parallel v_0) = H(6874ecd... \parallel 279f0ab...) = c8392d0 ...$
- $v_2 = H(v_1 \parallel p_1) = H(c8392d0... \parallel d401c1f...) = cb63aea... = R$

Was können wir nun mit dem Merkle Tree machen?

Merkle Proof ist effizient und offline fähig

Bei $n = 50.000$ Einträgen genügen $\lceil \log_2(n) \rceil = 16$ Hashwerte plus der Hash des Blatts, um die Merkle Root **offline** zu verifizieren – ganz **ohne Zugriff auf die Liste** oder **den restlichen Baum**.

Zur **Verifikation** genügt das **Speichern von 17 Hashwerten**, was beispielsweise **544 Byte** bei SHA-256 entspricht und problemlos in einen QR-Code passt (max. 2.953 Byte) [2].



3

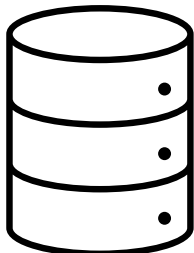
Merkle Trees - Praxiseinsatz

Wie können wir Merkle Trees nun für unser Konzert nutzen?

Erinnerung: Wir suchen nach einem Ansatz,

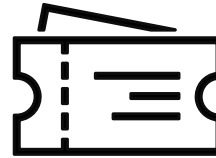
1. der **speichereffizient** ist (≈ 50.000 Einträge),
2. sich **offline** und **dezentral** verifizieren lässt,
3. auf allen Geräten **synchron** bleibt **ohne Online-Abgleich**,
4. der **manipulationssicher** ist (kein unbemerktes Ändern, Hinzufügen oder Löschen),
5. und **DSGVO-konform** arbeitet.

Idee:

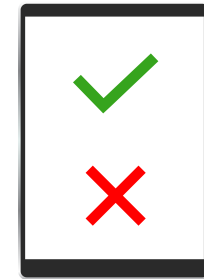


ID	Name
1	Alice
2	Bob
3	Charly
4	Dave
...	...
50.000	Zulu

Erstelle Merkle Tree mit R=
92bb49f46d6360c4552ced1f0738dd85402d
cbdcf52c10f1cea78f40



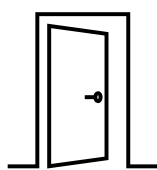
Erzeuge digitale Tickets (etwa QR
Code) mit: e_i , P , D des Besuchenden



Speichere Merkle Root
manipulationssicher auf den
Scannern und auf der Website.

Beispiel Merkle Proof: Ist Eve auf der List?

- $e_i = Eve,$
- $P = [7042062 \dots, \dots, cb63aea\dots]$
- $D = [1, \dots, 0]$



Ticketkontrolle 7

Merkle Root

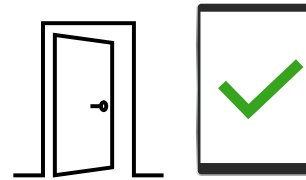
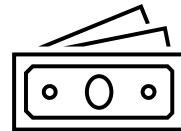
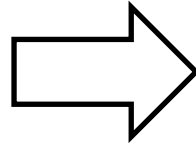
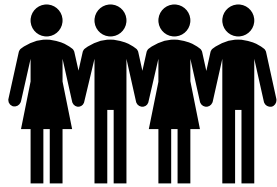
92bb49f46d6360c4552ced1f0738dd85402dcbdcf52c10f1cea78f40

$$v_0 = H(e_i), \quad v_j = \begin{cases} H(p_j \parallel v_{j-1}) & \text{falls } d_j = 0 \\ H(v_{j-1} \parallel p_j) & \text{falls } d_j = 1 \end{cases}$$

Der Beweis gilt als korrekt, wenn: $v_k = R$

- $v_0 = H(e_i) = f59b9c7\dots$
- $v_1 = H(v_0 \parallel 7042062\dots) = H(f59b9c7\dots \parallel 7042062\dots) = 0858160\dots$
- \dots
- $v_{16} = H(cb63aea\dots \parallel v_2) = H(cb63aea\dots \parallel d70d013\dots) = 92bb49f46d6360c4552ced1f0738dd85402dcbdcf52c10f1cea78f40 = R$

Übung: Ist dieses Szenario damit auch verhindert?



Ticketkontrolle 20

ID	Name
1	Alice
2	Bob
3	Charly
4	Dave
...	...
50.000	Zulu
50.001	<i>Thorsten</i>

4

Zusammenfassung und Ausblick

Was wir gesehen haben

Speichereffizient

Die Merkle Root fasst **beliebig viele Einträge** zu einem **einzigsten kompakten Hashwert** zusammen.

Offline verifizierbar

Die Korrektheit eines Eintrags kann **lokal** mit einem **Merkle Proof** überprüft werden [$\log_2(n)$] Hashwerte notwendig.

Manipulationssicher

Bereits eine **kleine Änderung** an einem Eintrag verändert die Merkle Root – **Manipulationen** werden **zuverlässig erkannt**.

Verteilbar und robust

Merkle Trees funktionieren auch dann, wenn Einträge auf **verschiedene Geräte** oder Knoten **verteilt** sind.



Wo Merkle Trees heute verwendet werden und wieso [2]

Blockchains & Kryptowährungen

Merkle-Proof belegt, dass Transaktion in einem Block steckt



Paketmanager & Softwareverteilung

Einzel-Paket per Hash verifizieren statt System-Download



Verteilte Dateisysteme (z. B. IPFS, Git)

Merkle Tree prüft Dateien & beschleunigt Sync



Zero-Knowledge-Proofs

Datengüte zeigen, ohne Inhalte preiszugeben



Limitationen

Kein Schutz der Vertraulichkeit

Merkle Trees sichern Integrität, aber nicht Vertraulichkeit.

Abhängigkeit von der Hashfunktion

Die gesamte Sicherheit hängt von Hashfunktion ab (MD5 ist daher keine gute Wahl)

Proof-Größe wächst mit $\log_2(n)$

Merkle Proofs sind effizient, aber nicht konstant klein (IoT ggf. ein Problem). Um einen Eintrag zu prüfen, benötigt man die gültige Merkle Root und $\lceil \log_2(n) \rceil$ Hashes.

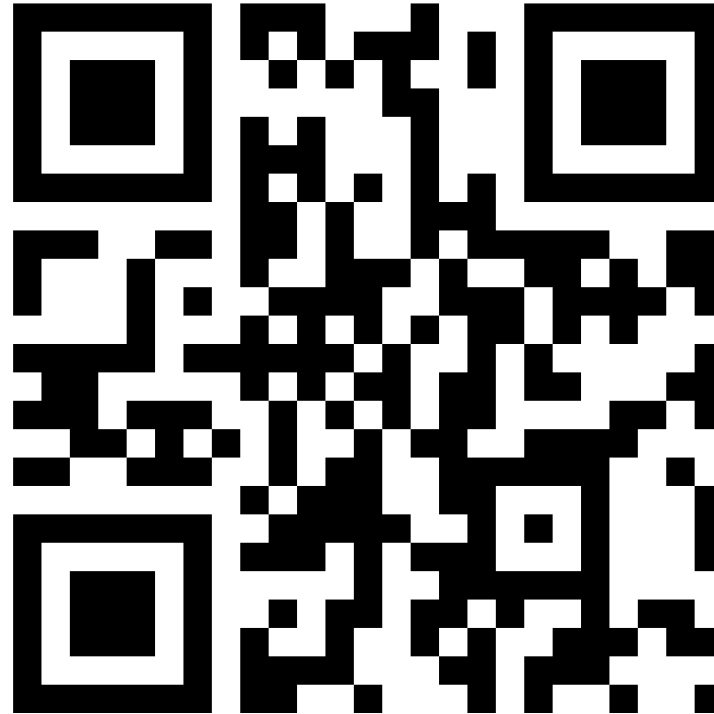
Keine inkrementellen Updates

Jede Änderung am Baum (z. B. neues Blatt) ändert die Merkle Root ggf. Overhead



Vielen Dank!

Live Demo:
<https://tinyurl.com/MerkleTree>



Gibt es Fragen?



Dr. Thorsten Weber

thorsten.weber88@web.de

Quellen

[1] Merkle, R. C. (1988).

A digital signature based on a conventional encryption function. In G. Brassard (Ed.), *Advances in Cryptology — CRYPTO '87* (pp. 369–378). Springer. https://doi.org/10.1007/3-540-48184-2_32

[2] Katz, J., & Lindell, Y. (2020). Introduction to modern cryptography (3rd ed.). CRC Press.