Crypto Workshop

Hashes und MACs

Christoph Egger
Dominik Paulus

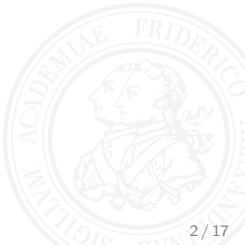
14. Mai 2018

Hash = Prüfsumme?

Prüfsumme

$$\neq$$

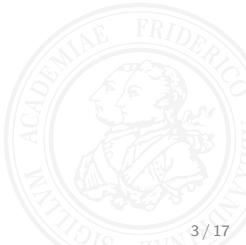
Kryptographische Hashfunktion



Hashes

Sicherheit

- *collision-resistance* – Es ist schwierig, zwei Eingaben mit dem gleichen Hashwert zu finden
- *(second) preimage resistance* – Wenn ich eine Prüfsumme (Eingabe) habe ist es schwierig, eine weitere EIngabe mit gleicher Prüfsumme zu finden



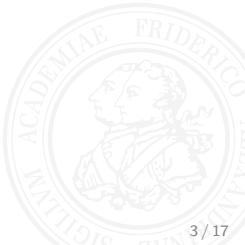
Hashes

Sicherheit

- *collision-resistance* – Es ist schwierig, zwei Eingaben mit dem gleichen Hashwert zu finden
- *(second) preimage resistance* – Wenn ich eine Prüfsumme (Eingabe) habe ist es schwierig, eine weitere EIngabe mit gleicher Prüfsumme zu finden

Eigenschaften

- Trivial erfüllt von $id(x) = x$



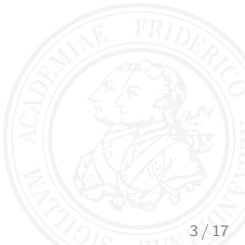
Hashes

Sicherheit

- *collision-resistance* – Es ist schwierig, zwei Eingaben mit dem gleichen Hashwert zu finden
- *(second) preimage resistance* – Wenn ich eine Prüfsumme (Eingabe) habe ist es schwierig, eine weitere EIngabe mit gleicher Prüfsumme zu finden

Eigenschaften

- Trivial erfüllt von $id(x) = x$
- Feste (kurze) Ausgabelänge



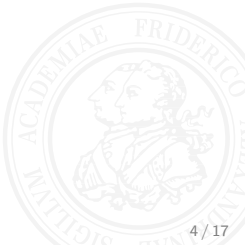
Komprimierungsfunktion

Ausgabe kürzen, ein Ansatz:

$$f : \{0,1\}^{2k} \to \{0,1\}^{k}$$

Muss auch nicht $2k$ sein, SHA256:

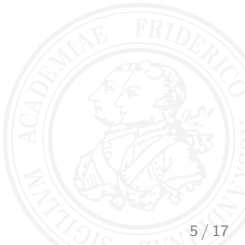
$$sha_{compress} : \{0,1\}^{768} \to \{0,1\}^{256}$$



Merkle-Damgård

Immer noch gesucht:

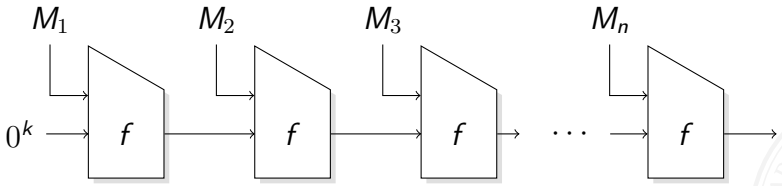
$$f : \{0,1\}^* \to \{0,1\}^k$$



Merkle-Damgård

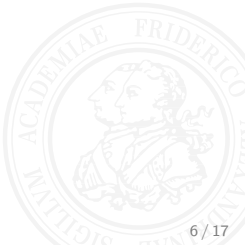
Immer noch gesucht:

$$f : \{0,1\}^* \to \{0,1\}^k$$



Verlängerungsangriff

Hashes zum Authentifizieren von Daten
Idee: $sha(x)$ sichert Integrität von x.



Verlängerungsangriff

Hashes zum Authentifizieren von Daten
Idee: $sha(x)$ sichert Integrität von x.

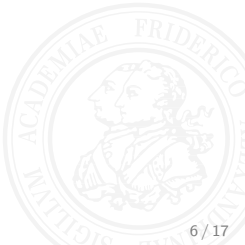
Aber: Jeder (auch der Angreifer) kann den Hash berechnen.
Idee: $sha(key \| x)$ – wer *key* kennt kann immer noch Integrität verifizieren aber der Angreifer kennt *key* nicht

Verlängerungsangriff

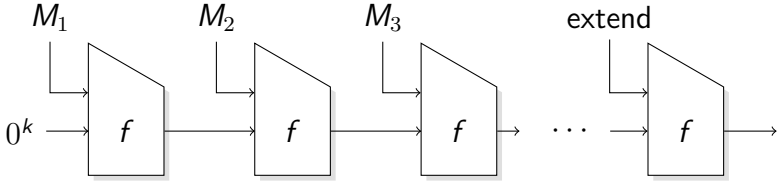
Hashes zum Authentifizieren von Daten
Idee: $sha(x)$ sichert Integrität von x.

Aber: Jeder (auch der Angreifer) kann den Hash berechnen.
Idee: $sha(key \| x)$ – wer *key* kennt kann immer noch Integrität verifizieren aber der Angreifer kennt *key* nicht

Ist das jetzt sicher?



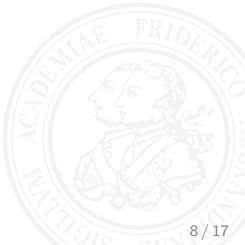
Merkle-Damgård



MAC

"Message Authentication Code"

"So was ähnliches wie eine Signatur"

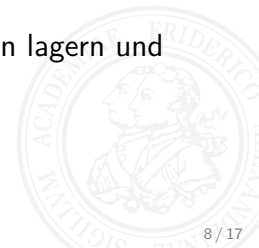


MAC

"Message Authentication Code"

"So was ähnliches wie eine Signatur"

- Parteien haben ein gemeinsames Geheimnis (key)
- Sicherstellen, dass Nachricht unverändert von einer Partei kommt, die das Geheimnis kennt
- Alle Parteien können durchaus die selbe sein – Nachricht extern lagern und sicherstellen, dass nichts verändert wurde
- TLS, SmartCard, TCP SYN-COOKIE, ...



HMAC

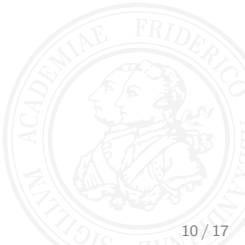
"MAC aus Hash bauen"

$$\mathrm{HMAC}(K, M) = H(K \oplus \mathrm{opad} \parallel H(K \oplus \mathrm{ipad} \parallel M))$$

Standardisiert (RFC2104), funktioniert für jede kryptographische Hash-Funktion (Annahme: Kollisionsresistenz)

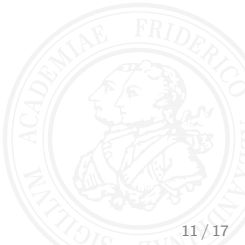
Fragen?

42

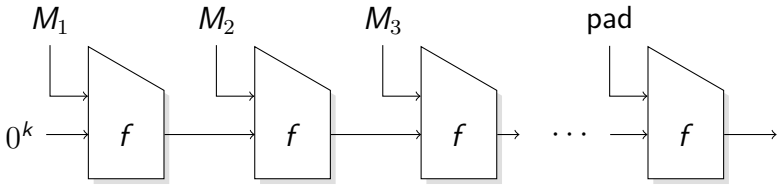


Hash Extension

- Funktioniert mit meinem kaputten SHA2 hervorragend
- Normalerweise: Sinnvolles Padding ist im Weg!



Merkle-Damgård



Hash Extension

- Funktioniert mit meinem kaputten SHA2 hervorragend

- Normalerweise: Sinnvolles Padding ist im Weg!
- Macht das ganze etwas frickeliger
- Trotzdem unsicher. Nehmt HMAC!

Unhiding Mac

```
CANDIDATES = ( 'Mihir Bellare', ... )

for cand in CANDIDATES:
    mac = telnet.read_until(b'\n')
                .rstrip().split(b' ')[2]
    candidates_by_mac[mac] = cand

for i in range(20):
    mac = telnet.read_until(b'\n')
                .rstrip().split(b' ')[2]
    telnet.write(candidates_by_mac[mac])
```

Abschliessend

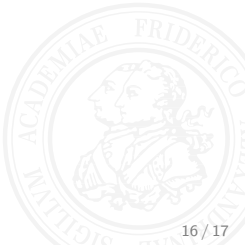
- Kollissionsresistenz sagt erstmal nichts über Hashes ähnlicher Eingaben

Abschliessend

- Kollissionsresistenz sagt erstmal nichts über Hashes ähnlicher Eingaben
- Hashes und MACs fürfen beliebiges über den Inhalt verraten! Gerade wenn die Eingabe Informationsarm ist, ist das auch praktisch ein Problem!

Was nun?

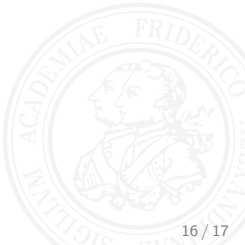
Verschlüsselung + MAC war eine schlechte Idee, ohne MAC sowieso. Wie mache ich das jetzt richtig?



Was nun?

Verschlüsselung + MAC war eine schlechte Idee, ohne MAC sowieso. Wie mache ich das jetzt richtig?

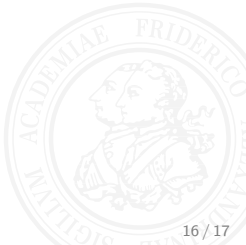
- MAC des Ciphertextes!



Was nun?

Verschlüsselung + MAC war eine schlechte Idee, ohne MAC sowieso. Wie mache ich das jetzt richtig?

- MAC des Ciphertextes!
- Keine Informationen mehr übrig $\Rightarrow$ MAC kann nichts verraten
- Wer den Ciphertext nicht ändern kann kann auch nicht verändern was beim entschlüsseln herauskommt

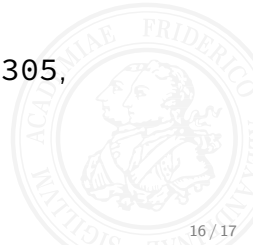


Was nun?

Verschlüsselung + MAC war eine schlechte Idee, ohne MAC sowieso. Wie mache ich das jetzt richtig?

- MAC des Ciphertextes!
- Keine Informationen mehr übrig $\Rightarrow$ MAC kann nichts verraten
- Wer den Ciphertext nicht ändern kann kann auch nicht verändern was beim entschlüsseln herauskommt

- "Authenticated Encryption" – `AES-GCM`, `ChaCha20/Poly1305`, AES-OCB, ...



Fragen?

42