Commitment-Schemes

Leonhard Applis

TH Nürnberg

XX.XX.XXXX

Table of Contents

Problem(s)

to do: What are the problems we need to adress

Commitments

- A **commits** to B
- B keeps commitment, is unable to read or process it
- A reveals to B
- B can verify the commitment

TODO: Image

Attributes

1. **Binding:** The Values Alice put in the Commitment cannot be changed after B recieved it
2. **XXX:** Bob cannot gain any information from the commitment itself

Additional for *real-life-applications*:

1. Bob's are able compare commitments
2. Commitments are *tradeable*

Applications

***Lean*-Login:**
You're able to access your Youtube-Account and favorites without login, but if you want to change your credentials you need to authenticate
this is done storing the commitment of your login in a cookie

JSON-Web-Tokens (JWT):
A payload (e.g. some account details) are encrypted to a commitment and passed to a third party.
You can verify yourself at the third-party revealing the commitment
this is done *automatic* via session or systemattributes

Table of Contents

1. Basics
2. Hash-Based
3. Binary
4. Discrete Log

Hash-Based Commitments

General Concept

1. Alice produces $h = Hash(m)$ and sends Bob h and $Hash$
2. Bob keeps h
3. Alice reveals herself by sending Bob m
4. Bob checks if $Hash(m) \equiv h$

Hash-Based Commitments

Problem: unlimited range - limited domain

Usually: Bob (and Eve) are not able to *guess* m from h and $Hash$

But: if the *plausible domain* of m is known, its possible for modern computers to brute force reveal your m

Example: Alice commits to Bob about the result of a soccer game Germany vs. Brazil.
Therefore she chooses a score of 0:7 and sends Bob $h = SHA_3(str(0:7))$ and the Hashfunction SHA_3
Eve catches the commitment and knows the context of the soccer game.
she can know try reasonable combinations of results from 0:0 up to 20:20.
She only needs to try $20 \cdot 20 = 400$ results

Hash-Based Commitments
Salting the Hash

Improved Concept:

- Alice chooses a random value s
- Alice produces $h = Hash(m, s)$ and sends h and $Hash$ to Bob
- Bob keeps h and $Hash$
- Alice reveals herself by sending bob m and s
- Bob checks if $Hash(m, s) \equiv h$

Table of Contents

1. Basics
2. Hash-Based
3. Binary
4. Discrete Log

Binary-Concept

Requirements and Definitions

Table of Contents

1. Basics
2. Hash-Based
3. Binary
4. Discrete Log

Discrete Logarithm

Requirements and Definitions