

Commitment-Schemes

Leonhard Applis

TH Nürnberg

20.12.2018

Table of Contents

Commitment
Schemes

Leonhard
Applis

Basics

Hash-
Based

Binary

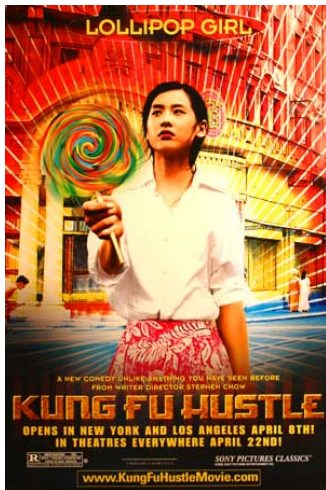
Discrete
Log

1 Basics

2 Hash-Based

3 Binary

4 Discrete Log



Plot Summary:

- 1 Villain destroys village and steals girls lollipop
- 2 Girls swears vengeance
- 3 Girl becomes Kung Fu - master
- 4 Girl finds the villain. Villain recognices her by the lollipop

Figure: Kung Fu Hustle - Lollipop Girl

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

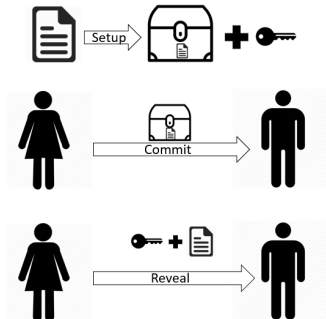


Figure: Commitments

- ① **Binding:** The Values Alice put in the Commitment cannot be changed after B received it
- ② **Hiding:** Bob cannot gain any information about the message from the commitment itself
- ③ **Viability:** If both parties follow the Protocol, Bob is always able to recover the committed value

Additional for *real-life-applications*:

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

Challenge and Response

You can setup your own anonymous challenges, leaving a commitment at Bob's. If someone shows up saying he's Alice, Bob challenges to reveal the commitment.

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 system attributes

Table of Contents

Commitment
Schemes

Leonhard
Applis

Basics

**Hash-
Based**

Binary

Discrete
Log

① Basics

② Hash-Based

③ Binary

④ Discrete Log

Hash-Based Commitments

General Concept

Commitment
Schemes

Leonhard
Applis

Basics

Hash-
Based

Binary

Discrete
Log

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

Important: NEVER use actual important data as message, you send it in cleartext in Step 3.

Hiding: because of the Hash-functions **Pre-Image resistance**, it's nearly impossible to find the message m from the hash. This holds true for any Bob and any Eve.

Binding: because of the Hash-functions **collision-resistance**, it's nearly impossible to find another message m with the same hash.

Hash-Based Commitments

Problem: unlimited range - limited domain

Commitm
Schemes

Leonhard
Applis

Basics

Hash-
Based

Binary

Discrete
Log

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

Improved Concept:

- 1 Alice chooses a random value s
- 2 Alice produces $h = \text{Hash}(m, s)$ and sends h and Hash to Bob
- 3 Bob keeps h and Hash
- 4 Alice reveals herself by sending bob m and s
- 5 Bob checks if $\text{Hash}(m, s) \equiv h$

Alice is anonymus. She never stated her name, used certificates, etc.
Alice can produce as many commitments for as many personas as she wants.

For increased security:

- commitments should be one-use only
- commitments should have a lifetime
- traded commitments to a third Party should revealed directly with first reveal
- messages must be chosen random

Table of Contents

Commitment
Schemes

Leonhard
Applis

Basics

Hash-
Based

Binary

Discrete
Log

1 Basics

2 Hash-Based

3 Binary

4 Discrete Log

Quadratic Residues

Definitions and Setup

Commitment
Schemes

Leonhard
Applis

Basics

Hash-
Based

Binary

Discrete
Log

Commitment to only 0,1 using quadratic residues.

A number n is **always** quadratic, if it's the product of two quadrats.

$$p^2 * q^2 = p * p * q * q = (p * q)^2$$

choose primes p, q and check for every element $x \in \mathbb{Z}_n^*, n = p * q$

The only way to check if x is quadratic in n , is to check if its quadratic for p and q !

Required: Legendre-Symbol $\left(\frac{x}{p}\right) = x^{p-1} \bmod(p)$

$\left(\frac{x}{p}\right)$	$\left(\frac{x}{q}\right)$	$\left(\frac{x}{n}\right)$	quadratic
1	1	1	yes
1	-1	-1	no
-1	1	-1	no
-1	-1	1	no

given X and guessing we have 75% cases where it's not quadratic.

given X and $\left(\frac{x}{n}\right) = 1$ we have 50:50 Quadratic:nonQuadratic

- ① Alice chooses primes p, q and an element $v \in \{x \in \mathbf{Z}_n | (\frac{x}{n}) = 1\}$
- ② Alice commits to a bit b by choosing a random number r and sending Bob: n, v and $c = r^2 \cdot v^b$
- ③ Bob verifies that $(\frac{v}{n}) = 1$ and keeps it
- ④ Alice reveals herself by sending Bob p, q, r, b
- ⑤ Bob verifies that p, q are primes, $n = pq$ and $c = r^2 \cdot v^b$

If A wants to commit a quadratic residue, she chooses a quadratic v and $b = 1$, therefore $r^2 \cdot v^1$ will be quadratic

If a wants to commit a non-quadratic value, she chooses a nonquadratic v and $b = 0$, therefore $r^2 \cdot v^0 = r^2$ which is quadratic

Table of Contents

Commitment
Schemes

Leonhard
Applis

Basics

Hash-
Based

Binary

Discrete
Log

1 Basics

2 Hash-Based

3 Binary

4 Discrete Log

Discrete Logarithm - Pedersen commitment scheme

Requirements and Definitions

Commitment
Schemes

Leonhard
Applis

Basics

Hash-
Based

Binary

Discrete
Log

Prerequisites: Bob needs to setup the environment for alice, by

- 1 choosing a large prime number p
- 2 choosing a smaller prime number $q \in \{1..p | q \div (p - 1) = 0\}$
- 3 choosing $g, v \in G_q \neq 1$
- 4 sending Alice p, q, g, v

Now Alice can *build* the exact same group and subgroup like Bob.
This is similiar to sending the hash-function.

- 1 Alice requests p, q, g, v from Bob. Alice check that q, p are primes, q divides $p-1$, that g and v are valid elements.
- 2 Alice chooses her message $m \in \{1..p\}$ and a random number $r \in \{1..q-1\}$
- 3 Alice sends $c = g^r v^m$ to Bob (**commit**)
- 4 Bob keeps $\langle Alice, c, \langle p, q, g, v \rangle \rangle$
- 5 Alice can reveal herself by sending r, m to Bob. Bob checks $c = g^r v^m$

Major:

- Commitments always contain random parts
- No collision possible (unlike Hashfunctions)

Minor:

- tuples are smaller to store than hashes
- p, q, g, v are easily changed/renewed (you could not renew hashfunctions)