Commitment

It allows one to commit to a chosen value while keeping it hidden to others.

However, it can be revealed at a later time when the one opens necessary parameter.

- Hiding; a given x and its commitment should be unrelatable.

  $\Rightarrow$ should <u>reveal no information</u> about x
- Binding; there is no way that different values can result in the same commitment.

  $\Rightarrow$ <u>cannot change</u> the value after committing

cf.

- Encryption
- Hash function

Pedersen Commitment [Ped92]

- $\mathsf{Setup}(1^\lambda)$;
  - Choose a large prime q and p such that $p = 2q + 1$
  - Choose $g \leftarrow \mathbb{Z}_q^*$
  - Choose $s \leftarrow \mathbb{Z}_q^*$ and compute $h := g^s$
  - Output (p, q, g, h)
- $\mathsf{Com}(x)$;
  - Choose $\gamma \leftarrow \mathbb{Z}_q^*$
  - Output $g^x h^\gamma$
- $\mathsf{Open}(\mathsf{Com}(x), x, \gamma)$;
  - Check whether c is equal to $g^x h^\gamma$ or not

<u>Note</u>

- Σ-Protocols: to prove knowledge of a committed value, equality of two committed values and so on.
- Since $g^{x_1} h^{\gamma_1} \cdot g^{x_2} h^{\gamma_2} = g^{x_1 + x_2} h^{\gamma_1 + \gamma_2}$,

$$\mathsf{Com}(x_1, \gamma_1) \cdot \mathsf{Com}(x_2, \gamma_2) = \mathsf{Com}(x_1 + x_2, \gamma_1 + \gamma_2)$$

- <u>Linear relatioships</u> among committed values can be shown through Pedersen commitments.
  - One could show that $y = ax + b$ for some public values a and b, given Com(x) and $\mathsf{Com}(y)$.

Example 1. (Σ-protocol for Pedersen Commitment)

Let p be a DL-secure prime and $g, h \leftarrow \mathbb{Z}_p^*$.

Prover and Verifier knows g, h and $y = g^x h^\gamma = \mathsf{Com}(x)$

Prover claims *to know a committed value x and γ.*

1. [Announcement]
   - Choose $s, t \leftarrow \mathbb{Z}_q^*$
   - Compute $a = g^s h^t$
   - Send a to Verifier
2. [Challenge]
   - Choose $c \leftarrow \mathbb{Z}_q^*$
   - Send c to Prover
3. [Response]
   - Compute $r_1 = cx + s$ and $r_2 = c\gamma + t$
   - Send r_1 and r_2 to Verifier
4. [Verification]
   - If $y^c \cdot a = g^{r_1} h^{r_2}$, then Verifier accepts
   - Otherwise, Verifier rejects

Example 2.

Let p be a DL-secure prime and $g, h \leftarrow \mathbb{Z}_p^*$.

Prover and Verifier knows g, h and $y_1 = g^{x_1} h^{\gamma_1} = \mathsf{Com}(x_1)$ and $y_2 = g^{x_2} h^{\gamma_2} = \mathsf{Com}(x_2)$.

Prover proves knowledge of x_1 and x_2 such that $x_2 = \alpha x_1 + \beta$, that is, proving knowledge of x such that

$$\{y_1 = \mathsf{Com}(x_1) \wedge y_2 = y_1^\alpha g^\beta\}$$

1. [Announcement]
   - Choose $s_1, s_2, t_1, t_2 \leftarrow \mathbb{Z}_q^*$
   - Compute $a_1 = g^{s_1} h^{t_1}$ and $a_2 = g^{s_2} h^{t_2}$
   - Send a_1 and a_2 to Verifier
2. [Challenge]
   - Choose $c_1, c_2 \leftarrow \mathbb{Z}_q^*$
   - Send c_1 and c_2 to Prover
3. [Response]
   - Compute $r_1 = c_1 x_1 + s_1, r_2 = c_1 \gamma + t_1, r_3 = c_2 x_2 + s_2$ and $r_4 = \alpha c_2 \gamma_1 + t_2$
   - Send r_1, r_2, r_3 and r_4 to Verifier
4. [Verification]
   - If $y_1^{c_1} \cdot a_1 = g^{r_1} h^{r_2}$ and $y_1^{c_2 \alpha} \cdot a_2 \cdot g^{c_2 \beta} = g^{r_3} h^{r_4}$, then Verifier accepts
   - Otherwise, Verifier rejects

zkSNARKs and zkSTARKs

zero-**k**nowledge of **S**uccinct **N**on-interactive **AR**guments of **K**nowledg**s**

zero-**k**nowledge of **S**calable **T**ransparent **AR**gument of **K**nowledge**s**

	Σ-Protocol	**zkSNARKs**	**zkSTARKs**
	no trusted setup	required a trusted setup	no trusted setup
Algebraic Statement	short proof size	large proof size	large proof size
Non-Algebraic Statement	large proof size	succinct proof size	larger than SNARKs
		verifiable computation	

- Σ-Protocol
    - zkBoo: [GMO16] @ USENIX'16,
    - Ligero: [AHIV17] @ CCS'17
- zkSNARK
    - Pinocchio: [PGHR13] @ S&P'13
    - Geppetto: [CFH+15] @ S&P'15
- Hybrid
    - [AGM18] @ CRYPTO'18

Privacy Problem in BTC

- Anonymity: *hiding identities* of sender and receiver
    - Monero uses a ring signature and a stealth address
    - ZCash uses a zero-knowledge proof
- Confidentially: *hiding the amount* transferred
    - Monero uses a confidential transaction (CT)
        - every transaction amount is hidden using a commitment to the amount
    - ZCash uses a commitment and a public key encryption

In CT, a zero-knowledge for range proof is used.

1. $\sum$ input $\geq \sum$ output
2. all transferred value ≥ 0

Current Proposals for CT-ZKP

- [PBF+]: large proof size or required a trusted setup
- SNARKs: required a trusted setup
- STARKs: large range proof

- Bulletproof: [BBB+18] @ S&P'18