

Commitment schemes

- An electronic way to temporarily hide a value that cannot be changed
 - Stage 1 (Commit)
 - Sender locks a message in a box and sends the locked box to another party called the Receiver
 - State 2 (Reveal)
 - the Sender proves to the Receiver that the message in the box is a certain message

Types of commitment

- Bit commitment
- Integer commitment
- String commitment

Security properties of commitment schemes

- Hiding
 - at the end of Stage 1, no adversarial receiver learns information about the committed value
- Binding
 - at the end of State 1, no adversarial sender can successfully convince reveal two different values in Stage 2

A broken commitment scheme

- Using encryption
 - Stage 1 (Commit)
 - the Sender generates a key k and sends $E_k[M]$ to the Receiver
 - State 2 (Reveal)
 - the Sender sends k to the Receiver, the Receiver can decrypt the message
- What is wrong using the above as a commitment scheme?

Formalizing Security Properties of Commitment schemes

- Two kinds of adversaries
 - those with infinite computation power and those with limited computation power
- Unconditional hiding
 - the commitment phase does not leak any information about the committed message, in the information theoretical sense (similar to perfect secrecy)
- Computational hiding
 - an adversary with limited computation power cannot learn anything about the committed message (similar to semantic security)

Formalizing Security Properties of Commitment schemes

- Unconditional binding
 - after the commitment phase, an infinite powerful adversary sender cannot reveal two different values
- Computational binding
 - after the commitment phase, an adversary with limited computation power cannot reveal two different values
- No commitment scheme can be both unconditional hiding and unconditional binding

Another (also broken) commitment scheme

- Using a one-way function H
 - Stage 1 (Commit)
 - the Sender sends $c=H(M)$ to the Receiver
 - State 2 (Reveal)
 - the Sender sends M to the Receiver, the Receiver verifies that $c=H(M)$
- What is wrong using this as a commitment scheme?

Modular Arithmetic

- Used to define a finite field
- $a = b \pmod n$ means that if a and b are divided by n they produce the same remainder
- $a \cdot b \pmod n$ can result in 0 even if a and b are not 0
- $a/b \pmod n$ is calculated by $a \cdot b^{-1} \pmod n$, where b^{-1} is the inverse of b , $b \cdot b^{-1} = 1 \pmod n$
 - $a/b = c \Leftrightarrow a = c \cdot b \Leftrightarrow a \cdot b^{-1} = c \cdot b \cdot b^{-1} = c \pmod n$

Commutative Ring

- Integers modulo n with addition (+) and multiplication (\cdot) form a commutative ring with following properties:
 - Additive associativity
 - $(a+b)+c = a+(b+c) \pmod n$
 - Additive commutativity
 - $a+b = b+a \pmod n$
 - Additive identity
 - $a+0 = 0+a = a \pmod n$
 - Additive inverse
 - $a+(-a) = 0 \pmod n$

Commutative Ring (cont.)

- Multiplicative associativity
 - $(a \cdot b) \cdot c = a \cdot (b \cdot c) \pmod n$
- Multiplicative commutativity
 - $a \cdot b = b \cdot a \pmod n$
- Multiplicative identity
 - $a \cdot 1 = 1 \cdot a = a \pmod n$
- Left and right distributivity
 - $a \cdot (b+c) = (a \cdot b) + (a \cdot c)$ and $(b+c) \cdot a = (b \cdot a) + (c \cdot a) \pmod n$

Finite Field

- A Field is a set that has the same properties as the commutative ring plus a multiplicative inverse for all elements but 0
 - $a \cdot a^{-1} = 1 \pmod n$, for all $a \neq 0$
- If n is constrained to be a prime number p then this forms a **Finite Field modulo p** denoted as **GF(p)** and all the normal laws associated with integer arithmetic work

Modular Exponentiation

- Many public-key encryption algorithms use modular exponentiation -- raising a number a (base) to some power b (exponent) mod p
- $c = a^b = a \cdot a \cdot \dots \cdot a \pmod p$

↑
b times

Multiplicative Group

- Fermat's Little Theorem
 - Let p be a prime, any integer a , $a^{p-1} = 1 \pmod{p}$
- Group generator
 - For a generator g , the sequence $g^0, g^1, g^2, \dots, g^{p-2}$ is a distinct sequence (all these $p-1$ elements are different). We call $\{g^0, g^1, g^2, \dots, g^{p-2}\}$ a multiplicative group with order $p-1$, also denote as Z_p^* ($=\{1, 2, \dots, p-1\}$).
 - Every operation is multiplication, $g^a \cdot g^b = g^{a+b}$
 - Consider the case $p=7, g=3$
 - It is easy to see that any number $y \in [1..p-1]$, there $\exists x$, such that $g^x = y \pmod{p}$

Multiplicative Group (cont.)

- Sub-group generator
 - For a sub-group generator g , consider the sequence g^0, g^1, g^2, \dots , let $q > 0$ be the first integer such that $g^q = 1$. We call $\{g^0, g^1, g^2, \dots, g^{q-1}\}$ a multiplicative group G_q , order- q subgroup of Z_p^* .
 - Consider $p=7, g=2, q=3$
 - $2^0, 2^1, 2^2, 2^3=1 \pmod{7}$

Discrete Logarithm Problem (DLP)

- Consider $GF(p)$ where p is a large prime. Given a multiplicative group (G, \cdot) , a generator g of G , and an element y in the group generated by g , denoted $\langle g \rangle$.
- DL problem is to find the unique integer x such that

$$g^x \bmod p = y$$

- x is the discrete logarithm
- DL problem is computational infeasible

Diffie-Hellman Problem in $GF(p)$

- Computational Diffie-Hellman (CDH) problem
 - Given g^a, g^b , compute g^{ab}
 - CDH is computational infeasible
- Decisional Diffie-Hellman (DDH) problem
 - Given g^a, g^b, y , decide whether $g^{ab} = y$
 - DDH is computational infeasible

Pedersen Commitment Scheme

- Setup
 - The receiver chooses two large primes p and q , such that $q|(p-1)$. Typically, p is 1024 bit, q is 160 bit. The receiver chooses a generator g of G_q , the order- q subgroup of Z_p^* , she also chooses secret a randomly from Z_q . Let $h = g^a \bmod p$. Values $\langle p, q, g, h \rangle$ are the public parameters and a is the private parameter.
- Commit
 - The domain of the committed value is Z_q . To commit an integer $x \in Z_q$, the sender chooses $r \in Z_q$, and computes $c = g^x h^r \bmod p$
- Open
 - To open a commitment, the sender reveal x and r , the receiver verifies whether $c = g^x h^r \bmod p$.

Pedersen Commitment Scheme (cont.)

- Unconditionally hiding
 - Given a commitment c , every value x is equally likely to be the value committed in c .
 - For example, given x, r , and any x' , there exists r' such that $g^x h^r = g^{x'} h^{r'}$, in fact $r = (x-x')a^{-1} + r' \pmod{q}$.
- Computationally binding
 - Suppose the sender open another value $x' \neq x$. That is, the sender find x' and r' such that $c = g^{x'} h^{r'} \pmod{p}$. Now the sender knows x, r, x' , and r' s.t., $g^x h^r = g^{x'} h^{r'} \pmod{p}$, the sender can compute $\log_g(h) = (x'-x) \cdot (r-r')^{-1}$. Assume DL is hard, the sender cannot open the commitment with another value.

Zero Knowledge Proofs

- A kind of interactive proof system
- Involves a prover and a verifier
- Proving without revealing any other information

Two Kinds of Zero-Knowledge Proofs

- ZK proof of a statement
 - convincing the verifier that a statement is true without yielding any other information
 - example of a statement, a propositional formula is satisfiable
- ZK proof of knowledge of a secret
 - convincing the verifier that one knows a secret, e.g., one knows the discrete logarithm $\log_g(y)$

Properties of Interactive Zero-Knowledge Proofs

– Completeness

- Given honest prover and honest verifier, the protocol succeeds with overwhelming probability

– Soundness

- no one who doesn't know the secret can convince the verifier with nonnegligible probability

– Zero knowledge

- the proof does not leak any additional information

Formalizing Soundness

- If a prover A can convince a verifier, then a knowledge extractor exists
 - a polynomial algorithm that given A can output the secret

Formalizing ZK property

- A simulator exists
 - taking what the verifier knows before the proof, can generate a communication transcript that is indistinguishable from one generated during ZK proofs
- Three kinds of indistinguishability
 - perfect (information theoretic)
 - Statistical
 - computational

Schnorr Protocol (ZK Proof of Discrete Log)

- System parameter: p, q, g
 - $q \mid (p-1)$ and g is an order q element in \mathbb{Z}_p^*
- Public identity: c
- Private authenticator: a where $c = g^{-a} \pmod p$
- Protocol
 1. P: picks random r in $[1..q]$, sends $d = g^r \pmod p$,
 2. V: sends random challenge e in $[1..2^t]$
 3. P: sends $y=r+ea \pmod q$
 4. V: accepts if $d = g^y c^e \pmod p$

Security of Schnorr Protocol - Soundness

- Probability of forge: $1/2^t$
 - The prover who does not know a can cheat by guess e
 - Set $d = c^e g^y$ at the first step
- We build a knowledge extractor as follows. Suppose the prover is challenged twice with on same c , first with e_1 , second with e_2 .
 - Send e_1 , receive y_1 such that $g^{y_1} c^{e_1} = d$
 - Send e_2 , receive y_2 such that $g^{y_2} c^{e_2} = d$
 - $g^{y_1 - y_2} = c^{e_2 - e_1}$, output $\log_g(c) = (y_1 - y_2) \cdot (e_2 - e_1)^{-1}$

Security of Schnorr Protocol – Zero knowledge

- Honest-verifier zero knowledge
- In general, not zero knowledge if e is large
 - Because through interaction, the verifier obtains solution (d,y,e) to equation $d = g^y c^e \pmod p$, which the verifier herself might not be able to compute

Pedersen Commitment – ZK Prove know how to open

- Public commitment $c = g^x h^r \pmod{p}$
- Private knowledge x, r
- Protocol:
 1. P: picks random y, s in $[1..q]$, sends $d = g^y h^s \pmod{p}$
 2. V: sends random challenge e in $[1..q]$
 3. P: sends $u = y + ex, v = s + er \pmod{q}$
 4. V: accepts if $g^u h^v = dc^e \pmod{p}$
- Security property – similar to Schnorr protocol

Bit Proof Protocol

- Let $\langle p, q, g, h \rangle$ be the public parameters of the Pedersen commitment scheme. Let $x \in \{0, 1\}$, $c = g^x h^r \pmod p$
- The prover proves to the verifier x is either 0 or 1 without revealing x
 - Note that $c = h^r$ or $c = gh^r$
 - The prover proves that she knows either $\log_h(c)$ or $\log_h(c/g)$
 - Recall if the prover can predict the challenge e , she can cheat by sending $d = h^y c^{-e}$
 - The prover uses Schnorr protocol to prove the one she knows, and to cheat the other one

Bit Proof Protocol (cont.)

- Case 1: $c=h^r$
 - P: choose w, z_1, e_1 from Z_q , $a_0=h^w$,
 $a_1=h^{z_1}(c/g)^{-e_1}$
 - P \rightarrow V: a_0, a_1
 - V \rightarrow P: e
 - P: $e_0 = e - e_1 \pmod q$, $z_0 = w + r \cdot e_0 \pmod q$
 - P \rightarrow V: z_0, z_1, e_0, e_1
 - V: verify $e = e_0 + e_1$, $h^{z_0} = a_0 \cdot c^{-e_0}$, $h^{z_1} = a_1 (c/g)^{-e_1}$

Bit Proof Protocol (cont.)

- Case 2: $c=gh^r$
 - P: choose w, z_0, e_0 from Z_q , $a_0=h^w$, $a_1=h^{z_0}c^{-e_0}$
 - P \rightarrow V: a_0, a_1
 - V \rightarrow P: e
 - P: $e_1 = e - e_0 \pmod q$, $z_1 = w + r \cdot e_1 \pmod q$
 - P \rightarrow V: z_0, z_1, e_0, e_1
 - V: verify $e = e_0 + e_1$, $h^{z_0} = a_0 \cdot c^{-e_0}$, $h^{z_1} = a_1 (c/g)^{-e_1}$

Security of Bit Proof Protocol

- Zero-knowledge
 - The verifier cannot distinguish whether the prover committed a 0 or 1, as what the prover sends in the two cases are drawn from the same distribution.
- Soundness
 - Bit proof protocol is a proof of partial knowledge