Commitment Schemes

LATTICE

Shanghai Jiao Tong University

May 21, 2019

Overview

Content

Intuition

A commitment scheme allows a party to commit to a message m by publishing a commitment σ, and this commitment can be opened at a later point in time, and the receiver will be convinced that the sender did not "change his mind".

- Binding: The binding property means that the sender cannot open a commitment σ to two different messages $m \neq m'$.
- Hiding: The hiding property means that the receiver cannot learn anything about the committed message m from the commitment σ.

Definition of Commitment Scheme

Definition (Commitment)

A triple of algorithms $(KGen, Com, Ver)$ is called a commitment scheme if it satisfies the following:

- $pk \leftarrow KGen(1^k)$
  On input 1^k, the key generation algorithm $KGen$ outputs a public commitment key pk.
- $(c, d) \leftarrow Com(m, pk)$
  The commitment algorithm Com takes as inputs a message m from a message space $\mathcal{M}$ and a commitment key pk, and outputs a commitment/opening pair (c, d).
- $0/1 \leftarrow Ver(pk, m, c, d)$
  The verification algorithm Ver takes a key pk, a message m, a commitment c and an opening d and outputs 1 or 0.

Properties of Commitment Scheme

Definition

The commitment scheme we construct satisfies the following security properties:

- Correctness: Ver evaluates to 1 whenever the inputs were computed by an honest party, i.e., $\Pr[Ver(pk, m, c, d) = 1;\ pk \xleftarrow{\$} KGen(1^k),\ m \in \mathcal{M},\ (c, d) \xleftarrow{\$} Com(m, pk)] = 1$
- Binding: With overwhelming probability over the choice of the public key $pk \xleftarrow{\$} KGen(1^k)$, no commitment c can be opened in two different ways, i.e., $(Ver(pk, m, c, d) = 1) \wedge (Ver(pk, m', c, d') = 1) \Rightarrow m = m'$
- Hiding: A commitment c hides the committed message with overwhelming probability over the choice of $pk \xleftarrow{\$} KGen(1^k)$, for every $m, m' \in \mathcal{M}$ and $(c, d) \xleftarrow{\$} Com(m, pk)$, $(c', d') \xleftarrow{\$} Com(m', pk)$ the distributions c and c' are indistinguishable.

Content

Two Types of Commitment Schemes

- Standard Commitment Schemes
  A standard commitment scheme protects against a PPT receiver and an all powerful sender. We say such a scheme is computationally hiding and information-theoretically binding.
- Perfect Commitment Schemes
  A perfect commitment scheme protects against a PPT sender and an all-powerful receiver. We say it is computationally binding and information-theoretically hiding.

It can be proved that commitment schemes which are both information-theoretically hiding and information-theoretically binding do not exist.

Standard Commitment Schemes

Definition

A standard commitment scheme consists of a pair of PPT algorithms (S, R) satisfying the following:

- Correctness: For all k and all $b \in \{0,1\}$:
  $\Pr[(com, dec) \leftarrow \mathcal{S}(1^k, b) : \mathcal{R}(1^k, com, dec) = b] = 1$.
- Binding: The following is negligible even for an all-powerful $\mathcal{S}^*$:
  $\Pr[(com, dec, dec') \leftarrow \mathcal{S}^*(1^k) : \mathcal{R}(1^k, com, dec) = 0 \wedge \mathcal{R}(1^k, com, dec') = 1]$.
- Hiding: The following is negligible for any PPT $\mathcal{R}^*$:
  $\left| \Pr[b \leftarrow \{0,1\}; (com, dec) \leftarrow \mathcal{S}(1^k, b) : \mathcal{R}^*(com) = b] - \frac{1}{2} \right|$

Perfect Commitment Schemes

Definition

A perfect commitment scheme consists of a pair of PPT algorithms (S, R) satisfying the following:

- Correctness: For all k and all $b \in \{0,1\}$:
  $\Pr[(com, dec) \leftarrow \mathcal{S}(1^k, b) : \mathcal{R}(1^k, com, dec) = b] = 1$.
- Binding: The following is negligible for any PPT $\mathcal{S}^*$:
  $\Pr[(com, dec, dec') \leftarrow \mathcal{S}^*(1^k) : \mathcal{R}(1^k, com, dec) = 0 \wedge \mathcal{R}(1^k, com, dec') = 1]$.
- Hiding: The following is negligible even for an all-powerful $\mathcal{R}^*$:
  $\left| \Pr[b \leftarrow \{0,1\}; (com, dec) \leftarrow \mathcal{S}(1^k, b) : \mathcal{R}^*(com) = b] - \frac{1}{2} \right|$

Content

Standard Commitment Scheme from Special PKE (1)

Let (Gen, E, D) be a public-key encryption scheme with perfect decryption (i.e., a decryption error never occurs).

Consider the following, two-phase protocol between a sender $\mathcal{S}$ on input $b \in \{0,1\}$ and a receiver $\mathcal{R}$.

- In the first phase, $\mathcal{S}$ runs $(pk, sk) \leftarrow Gen(1^k; rGen)$, computes $c = \mathcal{E}_{pk}(b, rE)$ and sends (pk, c) to $\mathcal{R}$.
- In the second phase, $\mathcal{S}$ sends $(b, rGen, rE)$ to the receiver.

Standard Commitment Scheme from Special PKE (2)

Theorem

The above protocol constitutes a standard commitment scheme.

Proof.

- Perfect Binding follows directly from the perfect decryption property of the encryption scheme.
- Computationally Hiding follows directly from the hiding property of the encryption scheme, if the sender is honest, then *rGen* and *rE* are chosen uniformly at random, and hence b is hidden.

□

Standard Commitment Based on One-Way TDP (1)

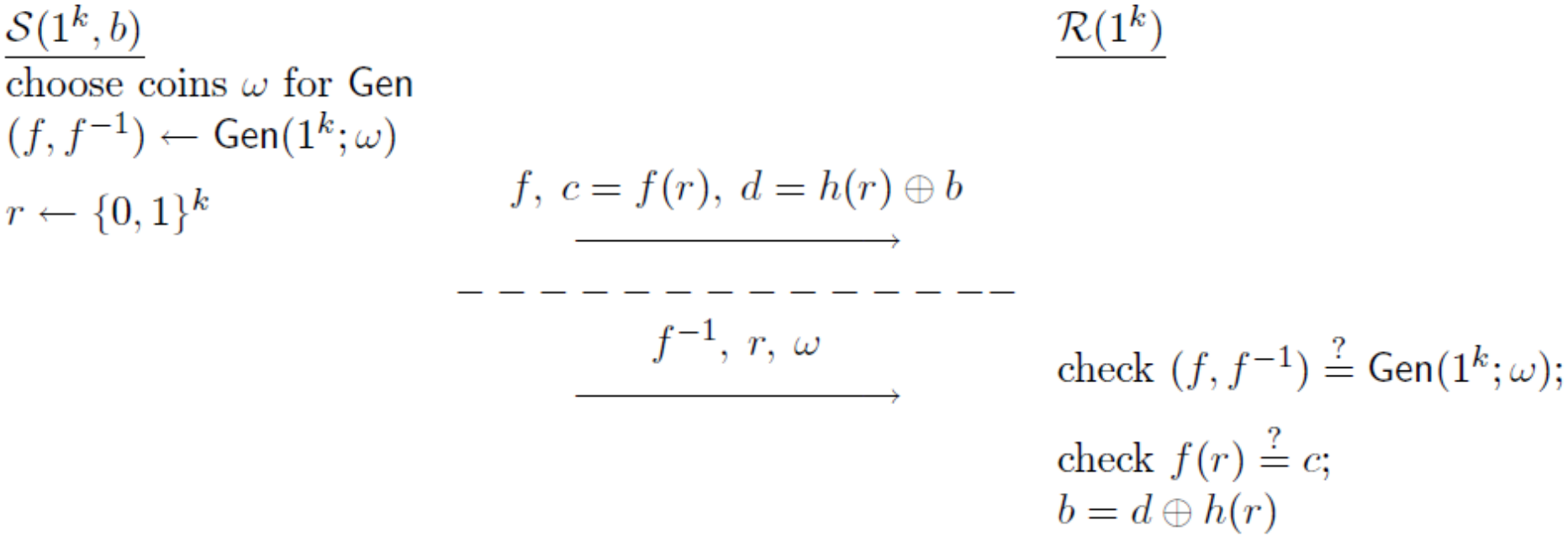

Figure: Standard Commitment Schemes Based on One-Way (Trapdoor) Permutations

The domain and range of the permutation f is $\{0,1\}^k$, $h(\cdot)$ is a hard-core bit of f.

Standard Commitment Based on One-Way TDP (2)

- Perfect Binding: We need to prove binding even for an all-powerful sender. Notice that the sender has to reveal the randomness ω used to generate f in the decommitment phase, and f sent in the first round is a permutation, Thus, c uniquely determines a value $r = f^{-1}(c)$, and this in turn uniquely determines $b = h(r) \oplus d$.
- Computationally Hiding: Since f is one-way, a PPT receiver cannot distinguish $h(r)$ from a random bit. Thus, $h(r)$ computationally hides the value of b.

Standard Commitment Based on One-Way Function (1)

Theorem

The existence of a one-way function implies the existence of a length-tripling pseudorandom generator (PRG).

Definition

$G = \{G_k : \{0,1\}^k \to \{0,1\}^{3k}\}_{k\in\mathbb{N}}$ is a pseudorandom generator (PRG) if the following is negligible for any PPT distinguisher D:

$$\left|\Pr[x \leftarrow \{0,1\}^{3k} : D(x) = 1] - \Pr[s \leftarrow \{0,1\}^k; x = G_k(s) : D(x) = 1]\right|.$$

In other words, $U_{3k} \approx G(U_k)$.

Standard Commitment Based on One-Way Function (2)

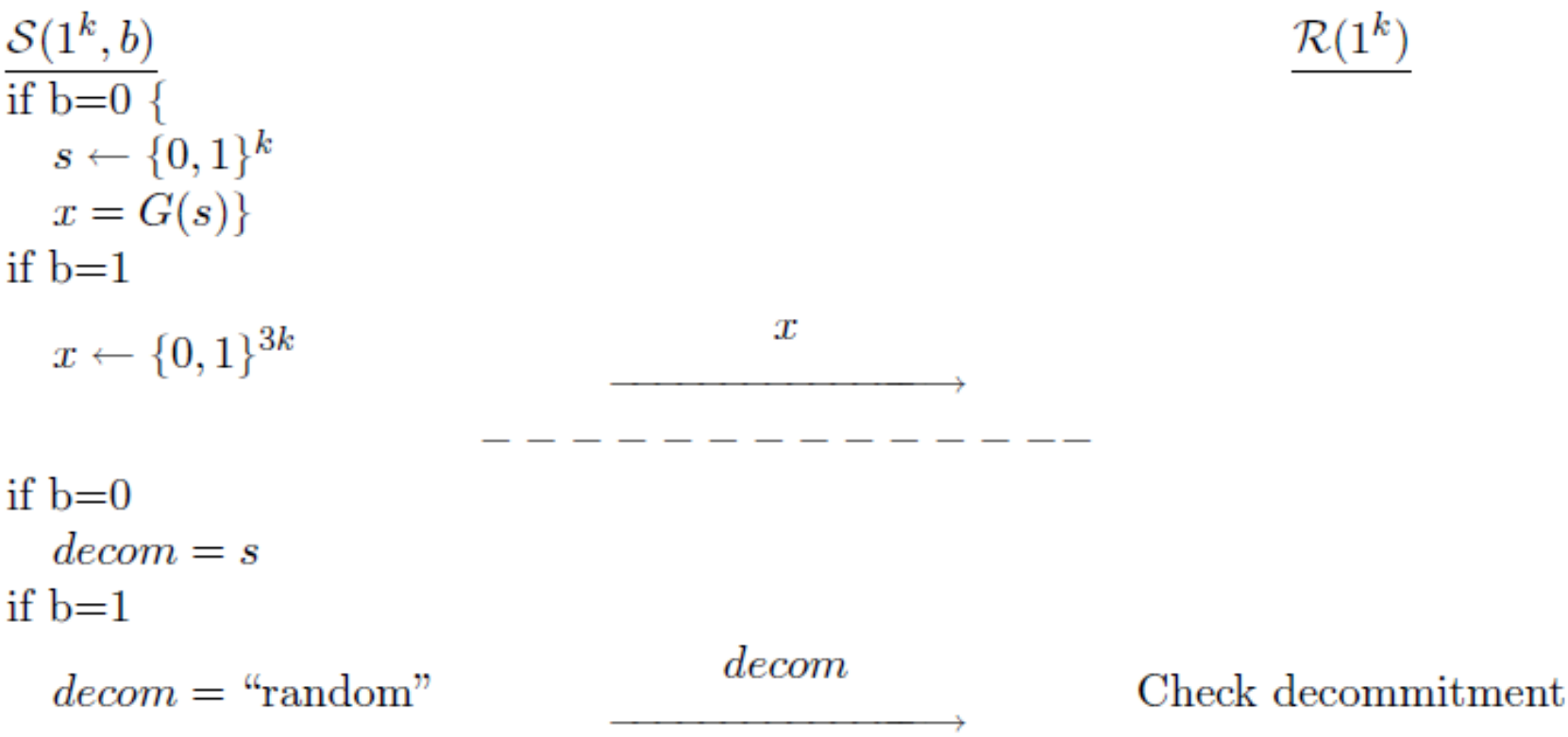

Figure: Naive Standard Commitment Schemes Based on One-Way Function

Standard Commitment Based on One-Way Function (3)

$\mathcal{S}$ sends a pseudorandom string to commit to "0", and a random string to commit to "1". To decommit, $\mathcal{S}$ sends the seed if x was pseudorandom (i.e., $b = 0$) or just says that x is "random" otherwise (i.e., $b = 1$).

- Computationally Hiding: $\surd$ This follows readily from the definition of a PRG.
- Perfect Binding: $\times$ A cheating $\mathcal{S}^*$ can send a pseudorandom string $x = G(s)$ and later decommit this to either a 0 (by sending s) or a 1 (by claiming that x is a random string).

Standard Commitment Based on One-Way Function (4)

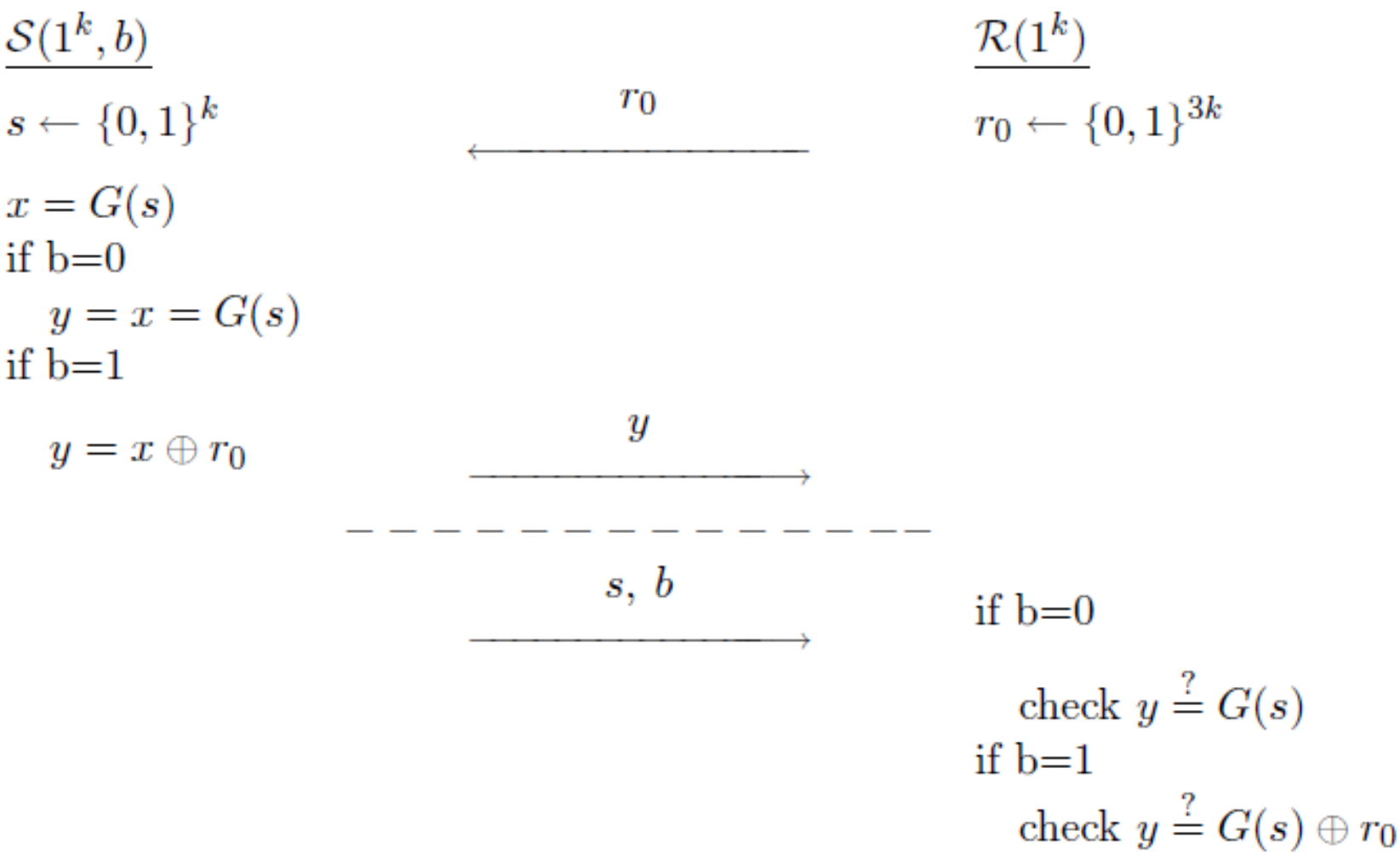

Figure: Standard Commitment Schemes Based on One-Way Function

Standard Commitment Based on One-Way Function (5)

Theorem (Computationally Hiding)

If a PPT cheating receiver $\mathcal{R}^*$ can distinguish a commitment to 0 from a commitment to 1 (before the decommitment phase), then we can use $\mathcal{R}^*$ to "break" the PRG.

Perfect Binding:

Note that $\mathcal{S}^*$ can only potentially cheat if $\mathcal{R}$ sends an r_0 for which there exist y, s, s' such that: $y = G(s)$ and $r_0 = y \oplus G(s')$ or, in other words, if there exist s, s' such that $G(s) \oplus G(s') = r_0$. Call r_0 with this property "bad".

Because s and s' are k-bit strings, there are at most 2^{2k} possible "bad" values r_0. Since r_0 is a uniformly-random $3k$-bit string, the probability of $\mathcal{R}$ choosing a "bad" r_0 is at most $2^{2k}/2^{3k} = 2^{-k}$, which is negligible.

Perfect Commitment Based on Discrete Logarithm (1)

The scheme allows a sender to commit to a string $x \in \{0,1\}^{k-1}$, not just a bit:

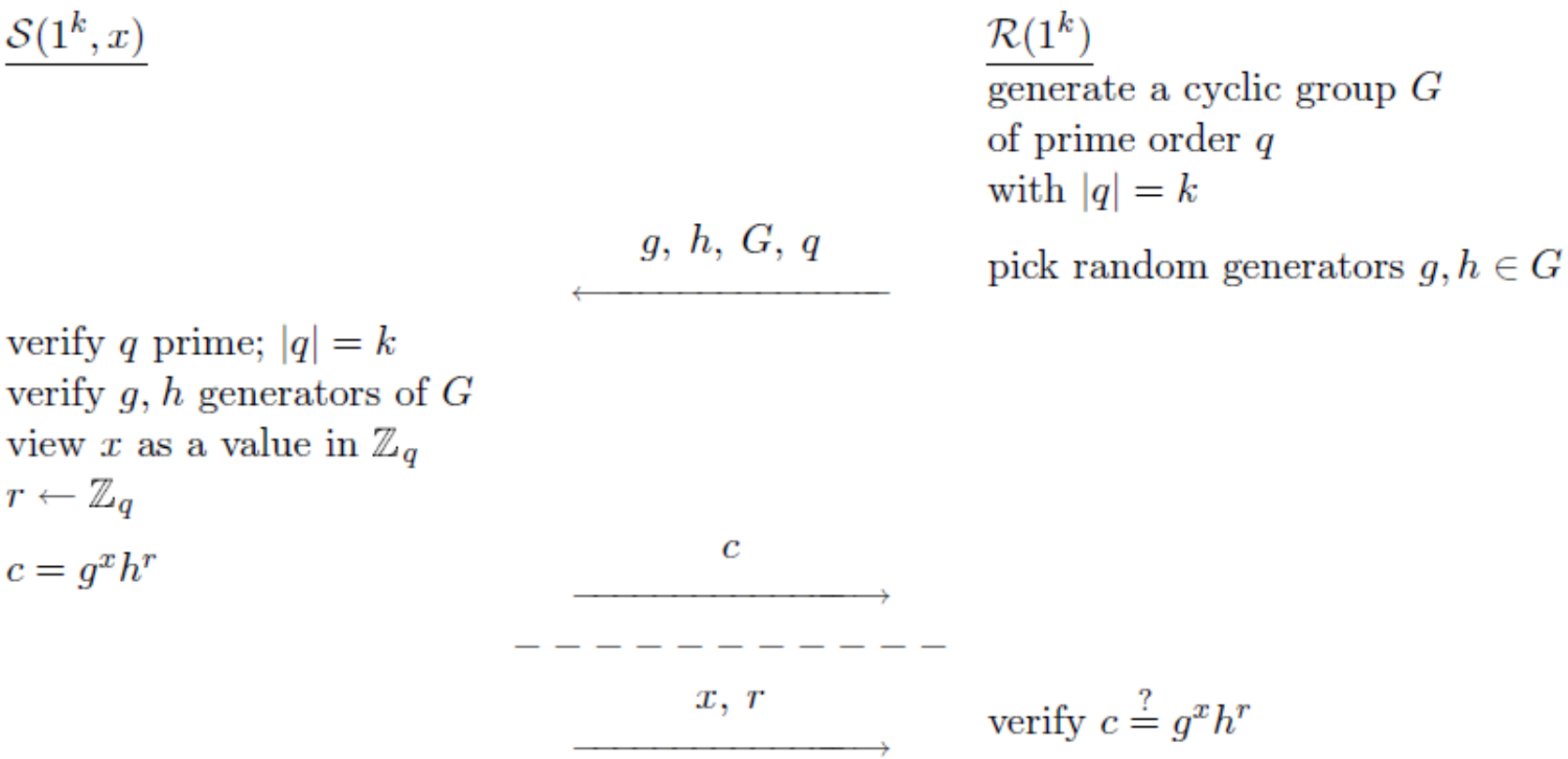

Figure: Perfect Commitment Schemes Based on Discrete Logarithm Assumption

Perfect Commitment Based on Discrete Logarithm (2)

- Perfect Hiding: Note that for any value c there are exactly q possible pairs (x, r) satisfying $g^x h^r = c$, one for each possible value of $x \in \mathbb{Z}_q$. The probability is exactly $1/|\mathbb{Z}_q| = 1/q = 1/2^k$. So, even an all-powerful $\mathcal{R}^*$ cannot tell which value of x is the one committed to by $\mathcal{S}$.
- Computationally Binding: If $\mathcal{S}^*$ is now able to decommit to (x, r) and (x', r'), with $x \neq x'$ and $g^x h^r = c = g^{x'} h^{r'}$, then $\mathcal{A}$ can compute the desired discrete logarithm by noting that

$$g^x h^r = g^{x'} h^{r'} \Leftrightarrow g^{x-x'} = h^{r'-r} \Leftrightarrow g^{(x-x')/(r'-r)} = h, \text{ or}$$
$$log_g h = (x - x')/(r' - r) \bmod q.$$

  Clearly, $\mathcal{A}$ runs in polynomial time if $\mathcal{S}^*$ does; also, if $\mathcal{S}^*$ decommits to two different values with probability $\varepsilon(k)$ then $\mathcal{A}$ correctly computes $log_g h$ with the same probability. Since this must be negligible under the discrete logarithm assumption, the binding property follows.

Perfect Commitment Based on RSA (1)

The scheme allows a sender to commit to a string $x \in \{0,1\}^k$, not just a bit:

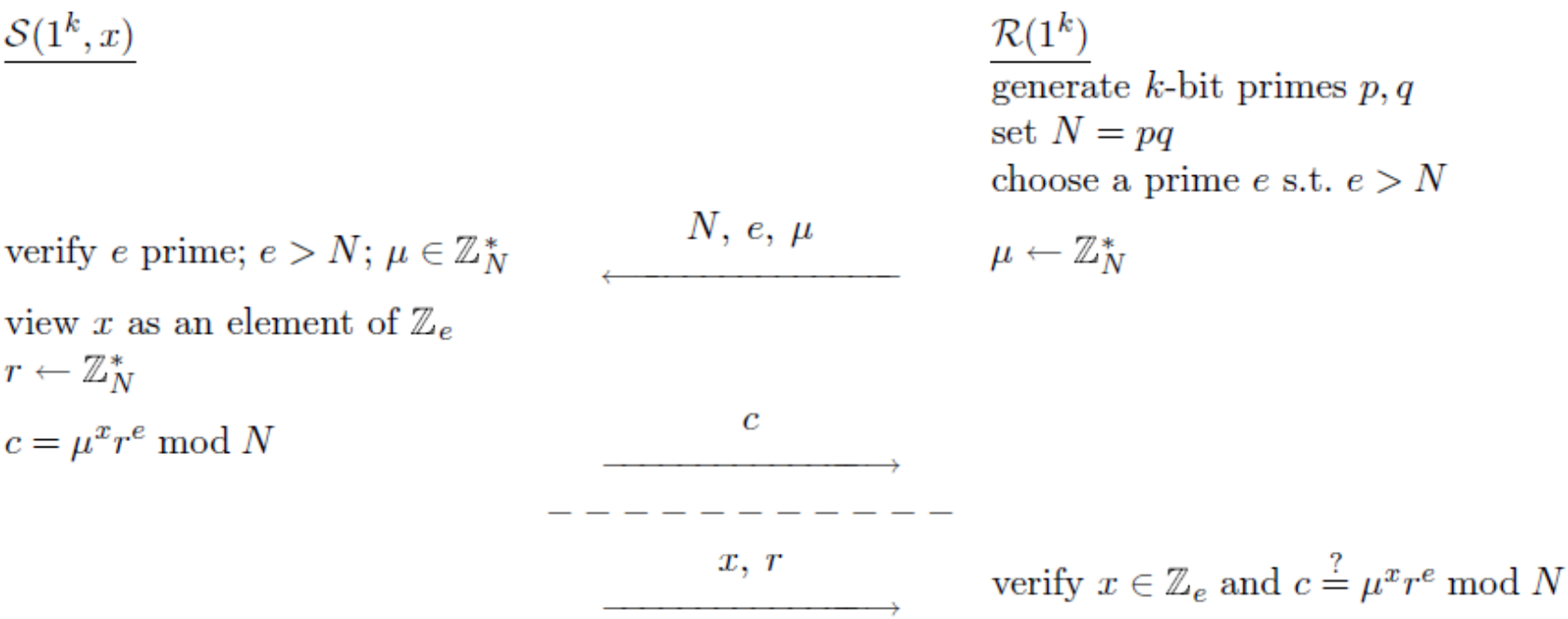

Figure: Perfect Commitment Schemes Based on RSA Assumption

Perfect Commitment Based on RSA (2)

Perfect Hiding:

The function $f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ defined by $f(x) = x^e \bmod N$ is a permutation. Since r is chosen uniformly at random in $\mathbb{Z}_N^*$, the value $f(r) = r^e \bmod N$ is uniformly distributed in $\mathbb{Z}_N^*$. Since $\mu \in \mathbb{Z}_N^*$ and hence $\mu^x \in \mathbb{Z}_N^*$, the product $c = \mu^x r^e$ is uniformly distributed in $\mathbb{Z}_N^*$ and reveals no information about x.

Another way to understand this is that for any commitment $c \in \mathbb{Z}_N^*$ and for any possible $x \in \mathbb{Z}_e$, there exists an $r \in \mathbb{Z}_N^*$ such that $c = \mu^x r^e \bmod N$. So no information about x is revealed.

Perfect Commitment Based on RSA (3)

Computationally Binding: we show how to use $\mathcal{S}^*$ to compute $\mu^{1/e}$. Simply send (N, e, μ) to $\mathcal{S}^*$ and assume it sends back $c \in \mathbb{Z}_N^*$ and can decommit to (x, r) and (x', r') such that $x \neq x'$, $\mu^x r^e = c = \mu^{x'}(r')^e \bmod N$, and both $x, x' \in \mathbb{Z}_e$. Without loss of generality assume $x > x'$. We know that: $\mu^x r^e = c = \mu^{x'}(r')^e \bmod N \Leftrightarrow \mu^{x-x'} = (r'/r)^e \bmod N$.

Let $\delta \stackrel{def}{=} x - x'$. Using the extended Euclidean algorithm, we can compute in polynomial time integers A, B such that $A\delta + Be = 1$ (over the integers).

$$\begin{aligned}\mu^1 = \mu^{A\delta+Be} &= (\mu^\delta)^A \mu^{Be} mod N \\ &= ((r'/r)^e)^A (\mu^B)^e mod N \\ &= ((r'/r)^A \mu^B)^e mod N\end{aligned}$$

and so $\mu^{1/e} = (r'/r)^A \mu^B \bmod N$.

This algorithm runs in polynomial time if $\mathcal{S}^*$ does, and outputs the desired inverse with the same probability that $\mathcal{S}^*$ can decommit to two different values. Since the former probability is negligible under the RSA assumption, the binding property follows.

References I

Jonathan Katz (2004)

CMSC 858K - Advanced Topics in Cryptography

M. Naor (1991)

Bit Commitment Using Pseudorandomness

Journal of Cryptology 4(2): 151-158 (1991).

The End