as the corresponding decommitment information). Thus, the sender can prove (to each of the other parties) in zero-knowledge that the message it is sending was indeed computed according to the original protocol.

A detailed description is provided in Section 7.4 (see also Section 7.5.4).

7.1.3.2 Passively-secure computation with "scrambled circuits"

The following technique refers mainly to two-party computation. Suppose that two parties, each having a private input, wish to obtain the value of a predetermined two-argument function evaluated at their corresponding inputs (i.e., we consider only functionalities of the form $(x, y) \mapsto (f(x, y), f(x, y))$). Further suppose that the two parties hold a circuit that computes the value of the function on inputs of the adequate length. The idea is to have one party construct an "scrambled" form of the circuit so that the other party can propagate encrypted values through the "scrambled gates" and obtain the output in the clear (while all intermediate values remain secret). Note that the roles of the two parties are not symmetric, and recall that we are describing a protocol that is secure (only) with respect to passive adversaries. An implementation of this idea proceeds as follows:

- *Constructing a "scrambled" circuit*: The first party constructs a "scrambled" form of the original circuit. The "scrambled" circuit consists of *pairs of encrypted secrets* that correspond to the wires of the original circuit and *gadgets* that correspond to the gates of the original circuit. The secrets associated with the wires entering a gate are used (in the gadget that corresponds to this gate) as keys in the encryption of the secrets associated with the wire exiting this gate. Furthermore, there is a *random correspondence* between each pair of secrets and the Boolean values (of the corresponding wire). That is, wire w is assigned a pair of secrets, denoted (s'_w, s''_w), and there is a random 1-1 mapping, denoted ν_w, between this pair and the pair of Boolean values (i.e., $\{\nu_w(s'_w), \nu_w(s''_w)\} = \{0, 1\}$).

  Each gadget is constructed such that knowledge of a secret that correspond to each wire entering the corresponding gate (in the circuit) yields a secret corresponding to the wire that exits this gate. Furthermore, the reconstruction of secrets using each gadget respects the functionality of the corresponding gate. For example, if one knows the secret that corresponds to the 1-value of one entry-wire and the secret that corresponds to the 0-value of the other entry-wire, and the gate is an OR-gate, then one obtains the secret that corresponds to the 1-value of exit-wire.

  Specifically, each gadget consists of 4 templets that are presented in a random order, where each templet corresponds to one of the 4 possible values of the two entry-wires. A templet may be merely a double encryption of the secret that corresponds to the appropriate output value, where the double encryption uses as keys the two secrets that correspond to the input values. That is, suppose a gate computing $f : \{0, 1\}^2 \to \{0, 1\}$ has input wires w_1

and w_2, and output wire w_3. Then, each of the four templets of this gate has the form $E_{s_{w_1}}(E_{s_{w_2}}(s_{w_3}))$, where $f(\nu_{w_1}(s_{w_1}), \nu_{w_2}(s_{w_2})) = \nu_{w_3}(s_{w_3})$.

- *Sending the "scrambled" circuit*: The first party sends the "scrambled" circuit to the second party. In addition, the first party sends to the second party the secrets that correspond to its own (i.e., the first party's) input bits (but not the values of these bits). The first party also reveals the correspondence between the pair of secrets associated with each output (i.e., circuit-output wire) and the Boolean values.[5] We stress that the random correspondence between the pair of secrets associated with each other wire and the Boolean values is kept secret (by the first party).
- *Oblivious Transfer of adequate secrets*: Next, the first party uses a (1-out-of-2) Oblivious Transfer protocol in order to hand the second party the secrets corresponding to the second party's input bits (without the first party learning anything about these bits).

  Loosely speaking, a 1-out-of-k Oblivious Transfer is a protocol enabling one party to obtain one of k secrets held by another party, without the second party learning which secret was obtained by the first party. That is, we refer to the two-party functionality

  $$(i, (s_1, ..., s_k)) \mapsto (s_i, \lambda) \qquad (7.1)$$

  where λ denotes the empty string.
- *Locally evaluating the "scrambled" circuit*: Finally, the second party "evaluates" the "scrambled" circuit gate-by-gate, starting from the top (circuit-input) gates (for which it knows one secret per each wire) and ending at the bottom (circuit-output) gates (for which, by construction, the correspondence of secrets to values is known). Thus, the second party obtains the output value of the circuit (but nothing else), and sends it to the first party.

For further details, see Section 7.7.4.

7.1.3.3 Passively-secure computation with shares

For any $m \geq 2$, suppose that m parties, each having a private input, wish to obtain the value of a predetermined m-argument function evaluated at their sequence of inputs. Further suppose that the parties hold a circuit that computes the value of the function on inputs of the adequate length, and that the circuit contains only AND and NOT gates. Again, the idea is to propagate information from the top (circuit-input) gates to the bottom (circuit-output) gates, but this time the information is different, and the propagation is done jointly by all parties. The idea is to share the value of each wire in the circuit such that all

[5] This can be done by providing, for each output wire, a succinct 2-partition (of all strings) that separates the two secrets associated with this wire.