## Basic Blum single-coin tossing using any commitment scheme

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | | P2 | |
| Input | None | Input | None |
| 1.Protocol | Choose random bits *b1* | 1.Protocol | Choose random bits *b2* |
|  | Commits to the single random bit *b1* using any commitment scheme |  | Participate in commitment |
|  |  |  | Send the random bit *b2* |
|  | Recieve random bit |  |  |
|  | Decommit to *b1* |  |  |
|  |  |  | Receive decommit |
| Output | XOR of the bits *b1*  and *b2* | Output | XOR of the bits *b1*  and *b2* |

## Basic Blum single-coin tossing using Pedersen commitment scheme

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | | P2 | |
| Input | 1*n* | Input | 1*n* |
| 1.Protocol | Choose random bits *b1*  Note that *b1* =*x* of commit | 1. Protocol | Choose random bits *b2* |
| 1.Commitment | Commits to the single random bit *b1* using any commitment scheme | 1.Commitment | Choose (G*, q, g*) where G is a group of order *q* with generator  *g* and *q >* 2*n* |
| Commitment |  | 2.Commitment | Choose a random *a ←* Z*q*, computes *α* = *ga* |
| Commitment |  | 3.Commitment | Send (G*, q, g, α*) |
| 2.Commitment | Receive (G*, q, g, α*) | Commitment |  |
| 3.Commitment | Verify that   1. G is a group of order *q*, 2. *g* is a generator 3. *α ∈* G. Then 4. If not all the above statements are true. Report error. | Commitment |  |
| 4.Commitment | choose a random *r ←* Z*q*, compute *c* = *gr · α b1* | Commitment |  |
|  | sends *c* |  |  |
|  |  | 4.Commitment | Receive *c* |
|  |  | 2.protocol | send the random bit *b2* |
| 2.protocol | Recieve random bit |  |  |
| 5.Commitment | Decommit to *b1* :  sends (*r, b1*) | Commitment |  |
|  |  | 5.Commitment | Receive decommit |
|  |  | 6.Commitment | verifies that *c* = *gr · α b1* |
| Output | XOR of the bits *b1*  and *b2* | Output | XOR of the bits *b1*  and *b2* |

# Oblivious Transfers

# Naor-Pinkas (using any DH group)

**PROTOCOL 7.2.1 (Private Oblivious Transfer *π*P**

**OT)**

|  |  |  |  |
| --- | --- | --- | --- |
| Receiver | | Sender | |
| Input | a bit *σ ∈ {*0*,* 1*}* | Input | *x*0*, x*1 of the same (arbitrary) length |
| Note |  |  | If actual inputs are not of the same length, report error. The calling protocol has to pad if they may not be the same length. |
| A. Input | * Security parameter 1*n* * Description of a group G of *prime order*, * A generator *g* for the group * The order of the group, *q*. * Probabilistic polynomial-time algorithm *V* | | |
| Note | The group can be chosen by *R* (receiver) if not given as auxiliary input. If R chooses the group, then it sends it to S in the first message. S must then check that it receives the description of a group of order q, where q is some prime. (If this is given by the dlog library then this can be an option. Otherwise, always use a fixed dlog group.) | | |
| 1.Protocol | choose *α, β, γ ←R {*1*, . . . , q}* and computes ¯*a* as follows:   1. If *σ* = 0 then   ¯*a* = (*gα, gβ, gαβ, gγ*).   1. If *σ* = 1 then   ¯*a* = (*gα, gβ, gγ, gαβ*). | Protocol |  |
| 2 | Send ¯*a* |  |  |
|  |  | 1 | Receive ¯*a* |
|  |  | 2 | Denote the tuple ¯*a* received by (*x, y, z*0*, z*1). |
|  |  | 3 | checks that all four values are in the group and that *z*0 *̸*= *z*1. |
|  |  | 4 | If the elements are not all in the group or if z0=z1, report error |
|  |  | 5 | choose random *u*0*, u*1*, v*0*, v*1 *←R {*1*, . . . , q}* and computes the following four values (all following operations in the group):  *w*0 = *xu*0 *· gv*0 *k*0 = (*z*0)*u*0 *· yv*0  *w*1 = *xu*1 *· gv*1 *k*1 = (*z*1)*u*1 *· yv*1 |
|  |  | 6 | Encrypts *x*0 under *k*0 and *x*1 under *k*1.  C0 = KDF(k0) XOR x0;  C1 = KDF(k1) XOR x1; |
|  |  | 7 | Sendthe pairs (*w*0*, c*0) and (*w*1*, c*1). |
| 3 | Receivethe pairs (*w*0*, c*0) and (*w*1*, c*1). |  |  |
| 4 | Check that w0,w1 are in the group and the c0,c1 are binary strings of the same length.  If not, report error.  Else, computes *kσ* = (*wσ*)*β* |  |  |
| Output | *xσ* = *cσ XOR KDF*(*kσ*). | Output | Nothing |

# Secure Pseudorandom Function Evaluation

PROTOCOL 7.6.3 (Private Pseudorandom Function Evaluation πP PRF)

|  |  |  |  |
| --- | --- | --- | --- |
| P1 | | P2 | |
| Input | key *k = (ga0 , a1, . . . , am)* where *a0, a1, . . . , am* ← *R Zq\**. | Input | *x* of length *m* |
| A Input | 1n and are given *G* – cyclic group, *q* prime and *g* generator. | | |
| Protocol |  | Protocol | Choose random bits *b2* |
|  | choose *m* random values *r1, . . . , rm ←R Zq\** |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |