PEP-DEC Protocols

10/17/23 : use VerifyDecryptionVariant, add test for ciphertext in Z_p^k.

EGDecryptTrust

Distributed Threshold Decryption and proof with trusted admin

Context

Principals:
    decrypting Guardians {DGi} where nd = |{DGi}| >= quota
    a trusted admin
    one or more verifiers V
Public input:
    Group parameters G = (p, q, g)
    joint public key K
    lagrange coefficients {wi} for the set of decrypting Guardians {DGi}
    ElGamalCiphertext (A, B)
Private input: for each DGi: Private key share pksi = P(i)

Output: T and ChaumPedersenProof(c, v)

Algorithm

1. DGi: Compute // 4
    test that A is an element of Z_p^k, ie A^q == 1; if not, abort decryption.
    Mi = A^pksi mod p
    ui ← Z∗q
    ai = g^ui mod p
    bi = A^ui mod p
  Send (Mi, ai, bi) to admin

2. admin: Compute
    M = Prod_di(Mi^wi) mod p // nd
    T = B/M mod p

3. admin: Compute
    a = Prod_di(ai) mod p
    b = Prod_di(bi) mod p
    c = H(He; 0x30, K, A, B, a, b, M) aka "collective challenge"
  Send challenge ci = (c * wi) mod q to each DGi

4. DGi: respond with vi = (ui − ci * pksi) mod q

5.a admin: Compute
    v = Sum_di(vi) mod q for i in {DGi}
    verify ChaumPedersenProof(c, v).verifyDecryption(g, K, A, B, T) is true // 4
    If true, skip 5.b

5.b admin: For each Gi, verify the response: // 4 * nd
  ai' = g^vi * GexpPi^ci mod p, where GexpPi is given below
  bi' = A^vi * Mi^ci mod p
 verify that ai' = ai and bi' = bi, Otherwise, reject.

6. admin
    publish (A, B, T, ChaumPedersenProof(c, v)) to BB

7. V:
    Verify ChaumPedersenProof(c, v).verifyDecryption(g, K, A, B, T) is true // 4

Notes:
* In this version, as much work as possible is done by a central "admin", rather than being done at each Gi.
* Step 1 tests that A is element of Z_p^k. This requirement will be added to 2.1 spec.
* Step 5 uses the ***VerifyDecryptionVariant***, which lets us skip 5.b if 5.a passes.
* Step 5.b uses a different form for verifying the individual Guardian responses, than CAKE distributed ElGamal decrypting protocol, due to the thresholding.
* The set of Guardians participating in the decryption is {DGi}, and the iteration over them is denoted Sum_di or Prod_di.
* The number of decrypting guardians = nd = |{DGi}| >= quorum.

Operation Count

each DG: 4 exp
admin: 4+nd (when 5.a verifies)
V: 4
DGs + admin = 4+5*nd

Regular exponentiation may be counted separately from accelerated exponentiation (which have g or K as the base). For decryption there are 3 regular for every accelerated exp.

GexpPi = g^P(i)

See eq 13, eq 74 of the EG 2.0 spec

Context

All guardians {Gi} (not just the decrypting guardians), n = |{Gi}|
Each guardian Gi has a secret polynomial Pi (x) = Sum_j(a_ij * x^j) mod q, j=0..quota-1
Each guardian Gi has public commitments K_ij = g^a_ij, j=0..quota-1
The election Polynomial P(x) is sum of guardian polynomials = P1(x) + P2(x) + .. +Pn(x)

Algorithm

g^P(i) = g raised to P(x), evaluated at the ith guardian's x coordinate
= Prod_i(Prod_j((K_ij)^i^j)) mod q), j = 0..quota-1), i = 1..n.

Operation count

n * n * quorum.
This value need only be computed once for each Guardian for all encryptions.

ChaumPedersenProof

General form

ChaumPedersenProof(c, v).verify(cons1; {const}, x, X, y, Y)
Proof that Prover knows s for two tuples (x, X=x^s) and (y, Y=y^s).
a = x^v * X^c mod p
b = y^v * Y^c mod p
verify that c = H(cons1; {const}, x, y, X, Y, a, b)

Operation count: 4 exp

EGDecrypt proof

ChaumPedersenProof(c, v).verifyDecryption(g, K, A, B, T)
Proof that Prover knows s for two tuples (g, K=g^s) and (A, M=A^s), where M = B / T.
a = g^v * K^c mod p
b = A^v * M^c mod p
verify that c = H(He; 0x30, K, A, B, a, b, M)

Operation count: 4 exp (2 regular, 2 accelerated)

EgkPepBlindTrust

Egk PEP with blinding Guardians separate from decrypting Guardians, and a trusted admin.

Context

Principals:

decrypting Guardians {DGi} where nd = |{DGi}| >= quota
blinding Guardians {BGj} where nb = |{BGj}| > 1
a trusted admin
one or more verifiers V

Public input:

Group parameters G = (p, q, g), Gi's public key share Ki for i = {1, · · · , n},
aggregated public key K
lagrange coefficients {wi} for the set of decrypting Guardians {DGi}
Enc(σj) = (αj, βj) for j ∈ {1, 2}.
Let α = α1/α2 mod p, β = β1/β2 mod p

Private input {Gi}: Private key share pksi = P(i)

Output: (IsEq, c, v, α, β, c′, v′, A, B, T)

Algorithm

1. all BGj in {BGj}: // 4
   (a) ξj ← Zq
   (b) Compute Aj = α^ξj mod p and Bj = β^ξj mod p.
   (c) uj ← Zq
   (d) Compute aj = α^uj mod p and bj = β^uj mod p
   (e) Send (Aj, Bj, aj, bj) to admin

2: admin:
   A = Prod_j(Aj)
   B = Prod_j(Bj)
   If (A == 1) or (B == 1), reject.

3. admin:
   a = Prod_j(aj)
   b = Prod_j(bj)
   c = H(cons0; {cons}, K, α, β, A, B, a, b)
   Send challenge c to each DGi

4. all BGj in {BGj}:
   respond to challenge with vj = uj − c * ξj
   send to admin

5.a admin:
   v = Sum_dj(vj)

verify if ChaumPedersenProof(c, v).verify(cons0; {cons1, K}, α, β, A, B). // 4
If true, can skip 5.b

5.b admin:
for each BGj, verify that aj = α^vj * Aj^c and bj = β^vj * Bj^c // 4 * nb

6. admin:
(a) decrypt (A, B): (T, ChaumPedersenProof(c',v')) = EGDecrypt(A, B) // 4+5*nd
(b) IsEq = (T == 1)
(c) Send (IsEq, c, v, α, β, c′, v′, A, B, T) to V and publish to BB.

7. V: read (IsEq, c, v, α, β, c′, v′, A, B, T) from BB
(a) verify if ChaumPedersenProof(c, v).verify(cons0; {cons1, K}, α, β, A, B). // 4
(b) verify if ChaumPedersenProof(c', v').verifyDecryption(g, K, A, B, T) // 4
(c) If T = 1, IsEq = 1 and (A, B) ≠ (1, 1), output "accept(equal)".
If T ≠ 1, IsEq = 0, output "accept(unequal)".
Otherwise, output "reject"

Notes:
* EGDecrypt uses the VerifyDecryptionVariant
* Step 5 uses the VerifyDecryptionVariant, which lets us skip 5.b if 5.a passes.

Operation count:

each BG: 4 exp
admin: EGDecrypt(4+5*nd) + 4 (when 5.a verifies)
admin+BGs: 8 + 4*nb + 5*nd

V: verify(4) + verifyDecryption(4) = 8

EGDecryptFull

Distributed Threshold Decryption without trusted admin, aka fully distributed

Context

Principals:
decrypting Guardians {DGi} where nd = |{DGi}| >= quota
one or more verifiers V
Public input:
Group parameters G = (p, q, g)
joint public key K
lagrange coefficients {wi} for the set of decrypting Guardians {DGi}
ElGamalCiphertext (A, B)
Private input: for each DGi: Private key share pksi = P(i)

Output: T and ChaumPedersenProof(c, v)

Algorithm

```
1. DGi: Compute               // 3
      Mi = A^pksi mod p
      ui ← Z∗q
      ai = g^ui mod p
      bi = A^ui mod p
   Send (Mi, ai, bi ) to other DGj

2. DGi: when received all (Mj, aj, bj):
      M = Prod_dj(Mj^wj) mod p    // nd
      T = B/M mod p
      a = Prod_di(ai) mod p
      b = Prod_di(bi) mod p
      c = H(He; 0x30, K, A, B, a, b, M) aka "collective challenge"
     vi = (ui − ci * pksi) mod q
    Send (ci, vi)  to other DGi

3. DGi: when received all (ci, vi), for each other Dj::            // 4 * (nd-1)
      aj' = g^vj * GexpPi^cj mod p
      bj' = A^vj * Mj^cj mod p
   verify that aj' = aj and bj' = bj, Otherwise, reject.

4. DGi: Compute
      v = Sum_di(vi) mod q for i in {DGi}
      publish (A, B, T, ChaumPedersenProof(c, v)) to BB

5. V: ChaumPedersenProof(c, v).verifyDecryption(g, K, A, B, T) is true    // 4
```

Operation Count

```
  Each DG: 3 + nd + 4*(nd-1) = 3 + 4*nd
  Total  = (3+4*nd)*nd per encryption

  With VerifyDecryptionVariant
   Each DG: 3 + nd + 4 = 7 + nd
   Total  = (7 + nd)*nd per encryption
```

EgkPepBlindFull

Egk PEP with blinding guardians separate from decrypting Guardians, no trusted admin.

Context

Principals:

decrypting Guardians {DGi} where nd = |{DGi}| >= quota
blinding Guardians {BGj} where nb = |{BGj}| > 1
one or more verifiers V

Public input:

Group parameters G = (p, q, g), Gi's public key share Ki for i = {1, · · · , n},
aggregated public key K
lagrange coefficients {wi} for the set of decrypting Guardians {DGi}
Enc(σj) = (αj, βj) for j ∈ {1, 2}.
Let α = α1/α2 mod p, β = β1/β2 mod p

Private input {Gi}: Private key share pksi = P(i)

Output: (IsEq, c, v, α, β, c′, v′, A, B, T)

Algorithm

1. all BGi in {BGj}: // 4
   (a) ξj ← Zq
   (b) Compute Aj = α^ξj mod p and Bj = β^ξj mod p.
   (c) uj ← Zq
   (d) Compute aj = α^uj mod p and bj = β^uj mod p
   (e) Send (Aj, Bj, aj, bj) to other BGj

2: BGi: when received all (Aj, Bj, aj, bj) :
   A = Prod_j(Aj)
   B = Prod_j(Bj)
   If (A == 1) or (B == 1), reject.
   a = Prod_j(aj)
   b = Prod_j(bj)
   c = H(cons0; cons1, K, α, β, A, B, a, b)
   vi = ui − c * ξi
   send vi to other BG

3. BGi:
   for each other BGj, verify that aj = α^vj * Aj^c and bj = β^vj * Bj^c // 4 * (nb-1)

4. BGi:
   (a) decrypt (A, B): (T, ChaumPedersenProof(c',v')) = EGDecryptFull(A, B)
   (b) v = Sum_dj(vj), IsEq = (T == 1)
   (c) Send (IsEq, c, v, α, β, c′, v′, A, B, T) to V and publish to BB.

5. V: read (IsEq, c, v, α, β, c′, v′, A, B, T) from BB
   (a) verify if ChaumPedersenProof(c, v).verify(cons0; {cons1, K}, α, β, A, B)   // 4
   (b) verify if ChaumPedersenProof(c', v').verifyDecryption(g, K, A, B, T) is true // 4
   (c) If T = 1, IsEq = 1 and (A, B) ≠ (1, 1), output "accept(equal)".
       If T ≠ 1, IsEq = 0, output "accept(unequal)".
       Otherwise, output "reject"

Operation count:

each BG: 4 + 4 * (nb-1) + EGDecryptFull((3+4*nd)*nd)
total = (4*nb + (3+4*nd)*nd)*nb

With VerifyDecryptionVariant:
each BG: 8 + EGDecryptFull((7 + nd)*nd)
total = (8 + (7+nd)*nd)*nb

EgkPepSimple

Egk PEP with decrypting Guardians, all other work done by a trusted admin

Context

Principals:
decrypting Guardians {DGi} where nd = |{DGi}| >= quota
A trusted admin
one or more verifiers V

Public input:
Group parameters G = (p, q, g), Gi's public key share Ki for i = {1, · · · , n},
aggregated public key K
lagrange coefficients {wi} for the set of decrypting Guardians {DGi}
Enc(σj) = (αj, βj) for j ∈ {1, 2}.
Let α = α1/α2 mod p, β = β1/β2 mod p

Private input {Gi}: Private key share pksi = P(i)

Output: (IsEq, c, v, α, β, c′, v′, A, B, T)

Algorithm

1. admin: // 4
   (a) ξ ← Zq
   (b) Compute A = α^ξ mod p and B = β^ξ mod p
   (c) u ← Zq

(d) Compute $a = \alpha^u \bmod p$ and $b = \beta^u \bmod p$
(e) $c = H(cons0; cons1, K, \alpha, \beta, A, B, a, b)$
(f) $v = u - c\xi$

2. admin:
(T, ChaumPedersenProof(c',v')) = EGDecrypt(A, B)
IsEq = (T == 1)
Send (IsEq, c, v, α, β, c′, v′, A, B, T) to BB.

3. admin and V: input (IsEq, c, v, α, β, c′, v′, A, B, T) from admin or BB
(a) verify if ChaumPedersenProof(c, v).verify(cons0; {cons1, K}, α, β, A, B)
Compute $a = \alpha^v * A^c$ and $b = \beta^v * B^c$
verify if $c = H(cons0 ; cons1 , K, \alpha, \beta, A, B, a, b)$
(b) verify if ChaumPedersenProof(c', v').verifyDecryption(g, K, A, B, T) is true
Compute $M = B/T \bmod p$, $a' = g^v * K^c \bmod p$ and $b' = A^v * M^c \bmod p$
verify if $v' \in Z_q$ and $c' = H(cons0; cons1, K, A, B, a', b', M)$.
(c) If T = 1, IsEq = 1 and (A, B) ≠ (1, 1), output “accept(equal)”.
If T ≠ 1, IsEq = 0, output “accept(unequal)”.
Otherwise, output “reject”.

Operation count:

Admin: 4 exp + EGDecrypt(8*nd) + verify(4) + verifyDecryption(4)
= 12 + 8*nd
V = 8

With VerifyDecryptionVariant:
Admin: 4 exp + EGDecrypt(4+4*nd) + verify(4) + verifyDecryption(4)
= 16 + 4*nd