Twisted ElGamal Encryption

Message m, encryption (public) key Y, decryption (secret) key sk

Two group elements $G, H \in \mathbb{G}$ (whose discrete log relations are unknown)

Key Generation

$$Y = sk^{-1}H$$

Encryption

Randomly sample r,

$$C = mG + rH$$

$$D = rY$$

Output (C, D) is the encrypted value. Note: C is the encrypted message, independent of the public key Y. D is the decryption handle, independent of the message being encrypted.

Decryption

$$mG = C - sk \cdot D$$

Then solve the discrete log to get m. Note: solving discrete log is slow, therefore we need the message size to be small.

Twisted ElGamal encryption is additive homomorphic. Consider two messages m_1 and m_2, encrypted with two different randomness r_1 and r_2

$$Enc(m_1, r_1) + Enc(m_2, r_2) = (C_1 + C_2, D_1 + D_2)$$

$$= ((m_1 + m_2)G + (r_1 + r_2)H, (r_1 + r_2)Y)$$

$$\equiv Enc(m_1 + m_2, r_1 + r_2)$$

Linear homomorphism means the messages can be added in the encrypted space. One important detail is that the randomness is also added. Which means if m_1 and m_2 are from different parties, neither one knows the randomness that encrypts the message $m_1 + m_2$. This does not matter for decrypting the message (and auditing), but it does matter for zero-knowledge proving of such a message (such as used in a range proof).

Randomness reuse: when generating ciphertext on the same message, we can reuse the randomness, such that the C portion is the same, only D is different with different encryption keys. This reduces the size of the transaction. For security analysis of randomness reuse, refer to the reference [1].

Σ-protocol

Σ-protocol is a method to prove some relations given some public inputs and secrets (witnesses) while preserving zero-knowledge on the secrets. At a high level it contains the following steps: 1. The prover generates commitment to some secret witnesses 2. The verifier generates a random challenge 3. The prover responds to the challenge by computing some "signatures" 4. The verifier is able to verify the original statements by verifying the "signatures", using the commitments instead of the witnesses, thus preserving zero-knowledge.

It is a protocol not a proof. A proof contains public inputs and witnesses and the relations that need to be proven.

Equality proof

Proves that a message m encrypts to C under randomness r, and multiple public keys under the same r can decrypt the same message. I.e. $C = mG + rH$ and for $i = 1, \dots, N$, $D_i = r \cdot Y_i$

(where N is typically $2 +$ num_auditors)

Public inputs: $(C, D_1, \dots, D_N, Y_1, \dots, Y_N, G, H)$, witnesses (m, r)

Steps of interactive proof

1. Prover picks random values a, b, computes $A = aG + bH$ and for $i = 1, \dots, N$, $B_i = b \cdot Y_i$ sends A, B_i to the verifier.
2. Verifier: generates a random challenge e and sends it to the prover
3. Prover computes $z_1 = a + e \cdot m$ and $z_2 = b + e \cdot r$, sends z_1, z_2 to the verifier
4. Verifier computes and evaluates the following equality relations:
   - $z_1 G + z_2 H \equiv A + eC$
   - For $i = 1, \dots, N$, $z_2 Y_i \equiv B_i + eD_i$

   If any of them is false, return `false`, otherwise return `true`

Valid amount proof

Proves that the amount m is correctly encoded, and is in the right range, i.e. $\mathbb{L}_{\text{right}} \subset \mathbb{L}_{\text{enc}} \bigwedge \mathbb{L}_{\text{range}}$.

$\mathbb{L}_{\text{range}}$ is the range proof that requires bulletproof, we will not describe it here. $\mathbb{L}_{\text{enc}}$ is a sigma proof very similar to the above $\mathbb{L}_{\text{equal}}$ with only one party.

Note: the range proof requires knowing the witness (m, r). Here we do, since the amount being encoded and the randomness is chosen by us.

Withdraw proof

Proves that the sender's new balance `b = m - v` where `m` is the initial balance, `v` is the sending amount, is within some range.

Setup The sender's initial balance was encrypted as $\tilde{C} = mG + \tilde{r}H$, $\tilde{D} = \tilde{r}Y$.

The sender encrypts the sending amount with zero randomness $V = vG$. It can do that because the sending amount will be unveiled anyway.

Apply the amount to the balance homomorphically, the new encrypted balance is $C = \tilde{C} - V$, decryption handle stays the same $D = \tilde{D}$.

If the sender knew the randomness $\tilde{r}$, then proving can be done with $\mathbb{L}_{\text{right}}$ above. However, the sender doesn't know that, because the randomness is the sum of all randomness from previous transactions (some

coming from other senders). So the sender needs to decrypt the amount first, then commit it with a new randomness r', in order to prove the new balance's range with Bulletproof.

Proof Description Public inputs: $(\tilde{C}, \tilde{D}, C', Y, G, H)$, witnesses (v, r, b, r', sk)

Statements that need to be proven:

1. The amount and balances adds up, and the sender knows the decryption key

$$\tilde{C} - vG = bG + sk \cdot \tilde{D}$$

2. The amount being committed is the same amount as above

$$C' = bG + r'H$$

3. The secret key corresponds to the sender's public key

$$Y = sk^{-1}H$$

Notes: 1. The withdraw proof does not involve any auditors, since the withdraw amount is not confidential 2. There is no "re-encryption" of the new balance i.e., in the end the sender still doesn't know the full randomness. The balance was decrypted and committed (with a new randomness) for the necessity of the range proof. 3. The full withdraw proof requires an additional range proof on the new balance.

TODO: describe the steps in the Sigma proof

Transfer proof

Proves a transfer amount is valid (does not underflow sender's balance or overflow receiver's) *and* the same transfer amount is correctly encrypted by all parties' public keys *and* the sender has the right secret key to decode its balance.

Setup Sender's initial balance m, sending amount v, sender's final balance $b = m - v$.

Sender's public key Y_s, receiver's public key Y_d, auditors' public keys $Y_1, ..., Y_N$

Sender's initial encrypted balance

$\tilde{C} = mG + \tilde{r}H$, $\tilde{D} = \tilde{r}Y$

The transfer amount needs to be encrypted once with randomness r^*, and $2 + N$ decryption handles

$C^* = vG + r^*H$, with sender handle $D_s^* = r^*Y_s$, receiver handle $D_d^* = r^*Y_d$, and auditor handles $D_1^* = r^*Y_1, ..., D_N^* = r^*Y_N$

Sender's new encrypted balance is derived homomorphically from the old encrypted balance and the encrypted amount (C, D)

$C = \tilde{C} - C^*$, decryption handle $D = \tilde{D} - D_s^*$.

The sender does not know the randomness so in order to prove its range, it needs to decrypt it to get the new balance, then commit to it with a new randomness r'

Decryption of the right amount: $\tilde{C} - C^* = bG + sk \cdot (\tilde{D} - D_s^*)$

The new commitment: $C' = bG + r'H$

Proof Description Public inputs $(\tilde{C}, \tilde{D}, C^*, D_s^*, D_d^*, D_1^*, ..., D_N^*, Y_s, Y_d, Y_1, ..., Y_N, C', G, H)$, witnesses (r^*, r', sk_s, v, b)

Statements that need to be proven in a transfer Sigma proof.

1. The right transfer amount v is committed with randomness r^*, and decryption handles for all parties contain the same randomness

$$C^* = vG + r^*H$$
$$D_s^* = r^*Y_s$$
$$D_d^* = r^*Y_d$$
$$D_i^* = r^*Y_i \text{ for } i = 1, ..., N$$

2. The balance b being committed with r' is same as the sender's new encrypted balance *and* the sender knows the secret key to decrypt it *and* the balance is the correct diff.

$$C' = bG + r'H$$
$$\tilde{C} - C^* = bG + sk \cdot (\tilde{D} - D_s^*)$$

3. The decryption key corresponds to sender's encryption key

$$Y_s = sk_s^{-1}H$$

Notes: 1. The full transfer proof requires two new range proofs - the transfer amount cannot overflow (fits within an `i64`) - the new balance of the sender is non-negative 2. We do not need to provide range proof for the recipient's new balance, because that's protected by the range proof on the amount, and the pending balance counter mechanism.

Normalization proof

Proves that before and after normalization the same balance quantity is being encrypted *and* each chunk in the new balance encrypts a value that fits within the range $(0 - 2^{16})$

Setup Assume without losing generality, the balance contains two chunks that are offset by 16 bits (initially when created, each chunk was 16-bits), and little-endian. At any given point (after many transfers), the balance ciphertext contains two chunks $(\tilde{C_0}, \tilde{C_1})$ and decryption handles $\tilde{D}$ and $\tilde{D_A}$ (for the auditor public key Y_A, assuming there is only one), under some randomness $\tilde{r}$ (not known to anyone).

After the balance is normalized, it contains two chunks (C_0, C_1) and decryption handles D and D_A under some new randomness r.

Proof Description Public inputs $(\tilde{C_0}, \tilde{C_1}, \tilde{D}, C_0, C_1, D, Y, G, H)$, witnesses (b_0, b_1, r, sk)

Statements to be proven:

1. The old encrypted amount equals the new encrypted amount:

$$\tilde{C_0} + 2^{16}\tilde{C_1} - C_0 - 2^{16}C_1 = 0 \cdot G + (1 + 2^{16})sk \cdot (\tilde{D} - D)$$

2. The new amount is correctly encoded with randomness r and the correct decryption handles are assigned, i.e.

$$C_0 = b_0G + rH$$
$$C_1 = b_1G + rH$$
$$D = rY$$
$$D_A = rY_A$$

3. The secret key corresponds to the account's public key

$$Y = sk^{-1}H$$

In addition, two range proofs are needed that both b_0 and b_1 fit within 16 bits.

References

- Twisted ElGamal: https://eprint.iacr.org/2019/319
- Bulletproofs: https://eprint.iacr.org/2017/1066