Constrain Satisfaction Problem CSP algorithms for multiagnet system often have some security issues, especially if one of the agent is untrusted. Agents' domain parameters are usually exchange insecurely, which makes such system vulnerable to wide range of cyber attacks. In this research we will primarily focus on a semi-honest adversary that try to learn the other agents' domain parameters that are used in CSP in order to bring the system to a compromised state. We will consider the case of a multiagent system (robots) that uses frequency hopping techniques and CSP to communicate securely in an untrusted environment. We will implement a cyber security system that is based on secure computation of hamming distance from COT. The proposed security protocol will make use of the secure multiparty computation and hamming distance to solve the CSP without leaking any domain parameters among the agents.

- **Inputs:**
  - P_1 inputs a n-bit string $X = (x_1, \ldots, x_n)$
  - P_2 inputs a n-bit string $Y = (y_1, \ldots, y_n)$
- **Output:**
  - 1^{st} Option: P_1 obtains $d_H(X,Y)$ and P_2 obtains nothing
  - 2^{nd} Option: P_2 obtains $d_H(X,Y)$ and P_1 obtains nothing
- **Protocol:**
  1. P_2 commits to all his bits y_i: he computes and publishes $Com(y_i, \chi_i)$ for each $i = 1 \ldots n$. He also proves, using π_1^2 proofs on the commitments, that $y_i = 0$ or $y_i = 1$.
  2. P_1 generates n random values $r_1, \ldots, r_n$, uniformly from the plaintext space of Com, and computes $R = \Sigma_{i=1}^{n} r_i$
  3. For each $i = 1, \ldots, n$, P_1 computes $(a_i, b_i) = (r_i + x_i, r_i + \bar{x_i})$ and commits to a_i and b_i. He computes and publishes $(A_i = Com(a_i, \alpha_i))_{i=1,\ldots,n}$ and $(B_i = Com(b_i, \beta_i))_{i=1,\ldots,n}$
  4. P_1 proves to P_2, using π_1^2 proofs on the commitments, that $|b_i - a_i| = 1$, for each $i = 1, \ldots, n$.
  5. For each $i = 1, \ldots, n$, P_1 and P_2 engage in a COT where
     - P_1 acts as the sender and P_2 as the receiver.
     - P_2's selection bit is y_i.
     - P_1's input is (a_i, b_i).
     - The output obtained by P_2 is $t_i = r_i + (x_i \oplus y_i)$ and τ_i.
     - Both parties obtain $C_i = Com(t_i, \tau_i)$
  6. P_2 computes $T = \Sigma_{i=1}^{n} t_i$,
  7. 1^{st} Option:
     (a) P_2 computes $C = Com(T, \tau) = C_1 \odot \ldots \odot C_n$
     (b) P_2 sends T and a zero-knowledge proof that C commits to T to P_1
     (c) P_1 computes $C = C_1 \odot \ldots \odot C_n$ and checks the proof.
     (d) P_1 computes and outputs $T - R$

     2^{nd} Option:
     (a) P_1 computes $K = Com(2R+n, \rho) = A_1 \odot \ldots \odot A_n \odot B_1 \odot \ldots \odot B_n$
     (b) P_1 sends R and a zero-knowledge proof that K commits to $2R+n$ to P_2
     (c) P_2 computes $K = A_1 \odot \ldots \odot A_n \odot B_1 \odot \ldots \odot B_n$ and checks that $K = Com(2R + n, \rho)$.
     (d) P_2 computes and outputs $T - R$

Fully Secure Scheme based on COT

- Committed *Com* in the algorithm is based on the ElGamal and Paillier Encryption/Decryption
- Zero-knowledge proofs π_1^2 is based on the following schemes:

Let E be Paillier encryption with public key (n, g).

The inputs of the prover are an element $x \in \mathbb{Z}_n$ and a random $r \in \mathbb{Z}_{n^2}^*$. The common inputs of the prover and the verifier are x_1, x_2 and $E(x, r) = g^x r^n \mod n^2$. We assume, w.l.o.g. that $x = x_1$. Let k denote the bit-length of n and $t = k/2$. Prover P and Verifier V proceed as follows:

1. P and V compute $u_1 = E(x)/g^{x_1} (= r^n)$ and $u_2 = E(x)/g^{x_2}$
2. P picks a random $z_2 \in_R \mathbb{Z}_{n^2}^*$, a random k-bit number e_2 and sets $a_2 = z_2^n u_2^{-e_2} \mod n^2$.
3. P picks a random $r_1 \in \mathbb{Z}_{n^2}$ and sets $a_1 = r_1^n \mod n^2$
4. P sends a_1 and a_2 to V
5. V chooses a random t-bit number s and sends it to P
6. P computes $e_1 = s - e_2 \mod 2^t$ and $z_1 = r_1 r^{e_1} \mod n^2$.
7. P sends e_1, z_1, e_2, z_2 to V
8. V checks that $s = e_1 + e_2 \mod 2^t$, $z_1^n = a_1 u_1^{e_1} \mod n^2$ and $z_2^n = a_2 u_2^{e_2} \mod n^2$ and accepts if and only if all checks succeed.

Let E be an ElGamal encryption with public key h in a group G of order q and generator g.

The inputs of the prover are an element x and a random r. The common inputs of the prover and the verifier are x_1, x_2 and $E(x, r) = (g^r, g^x h^r) = (b_1, b_2)$. We assume, w.l.o.g. that $x = x_1$. Prover P and Verifier V proceed as follows:

1. P and V compute $u_1 = b_2/g^{x_1} (= h^r)$ and $u_2 = b_2/g^{x_2}$
2. P picks random $v_1, v_2, c_2 \in_R \mathbb{Z}_q$ and computes $t_1 = h^{v_1}$ and $t_2 = u_2^{c_2} h^{v_2}$.
3. P sends t_1 and t_2 to V
4. V picks a random $c \in_R \mathbb{Z}_q$ and sends it to V
5. P computes $c_1 = c - c_2$, $r_1 = v_1 - c_1.r$ and $r_2 = v_2$
6. P sends c_1, r_1, c_2, r_2 to V
7. V checks that $c = c1 + c_2$, $t_1 = u_1^{c_1} h^{r_1}$ and $t_2 = u_2^{c_2} h^{r_2}$ and accepts if and only if all checks succeed.

Basic Scheme Based on OT_1^2

– **Inputs:**
  - P_1 inputs a n-bit string $X = (x_1, \ldots, x_n)$
  - P_2 inputs a n-bit string $Y = (y_1, \ldots, y_n)$

– **Output:**
  - 1^{st} Option: P_1 obtains $d_H(X, Y)$ and P_2 obtains nothing
  - 2^{nd} Option: P_2 obtains $d_H(X, Y)$ and P_1 obtains nothing

– **Protocol:**

1. P_1 generates n random values $r_1, \ldots, r_n \in_R \mathbb{Z}_{n+1}$ and computes $R = \Sigma_{i=1}^{n} r_i$
2. For each $i = 1, \ldots, n$, P_1 and P_2 engage in a OT_1^2 where
   - P_1 acts as the sender and P_2 as the receiver.
   - P_2's selection bit is y_i.
   - P_1's input is $(r_i + x_i, r_i + \bar{x_i})$.
   - The output obtained by P_2 is consequently $t_i = r_i + (x_i \oplus y_i)$.
3. P_2 computes $T = \Sigma_{i=1}^{n} t_i$
4. 1^{st} Option:
   (a) P_2 sends T to P_1
   (b) P_1 computes and outputs $T - R$
   2^{nd} Option:
   (a) P_1 sends R to P_2
   (b) P_2 computes and outputs $T - R$