# Distributed Merkle's Puzzles

**Original Merkle's Puzzles Idea**

1. Alice(A) and Bob(B) want to create a symmetric secret key ($K_{AB}$) between them. Adversary(Adv) wants to access the secret key by breaking the key exchanging algorithm..
2. A, B and Adv have the access to the random oracle H which is basically a Hash function.
3. There are N entries in the Hash function.
4. A picks up random $\sqrt{N}$ entries $(x_1, x_2, \dots, x_n)$ and calculate their hash value $(H(x_1), H(x_2), \dots, H(x_n))$.
5. A sends these hash values to B.
6. B picks up random $\sqrt{N}$ entries $(y_1, y_2, \dots, y_n)$ and calculate their hash values $(H(y_1), H(y_2), \dots, H(y_n))$.
7. Using Birhtday's Paradox, there is a high probability that some $H(x_i) = H(y_j)$ for some $x_i, y_j$. Here properties of hash functions make the collision (different inputs match to the same output) very unlikely.
8. B sends that $H(x_i)$ to the A. Now Bob and Alice have the same key with high probability.
9. Now the query complexity (Total number of queries to the hash function) for both A and B is $O(\sqrt{N})$ and communication complexity (Total number of entries communicated) for A is $O(\sqrt{N})$ and for B is $O(1)$.
10. Adv knows $H(x_i)$. But to know the $x_i$, Adv has to calculate the hash of all the N values. Hence, Adv query complexity is O(N).
11. In this protocol, we get Quadratic security after doing Linear work.
12. So the Security/Work ratio is O(N).

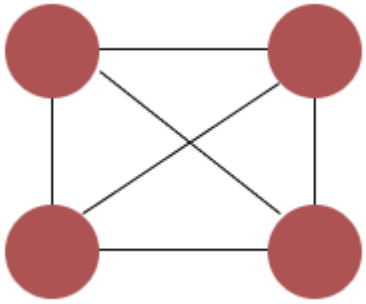
**Distributed Key Agreement**

In the paper, the aim is to establish a distributed key agreement among M players. Here all M players are connected to each other, similar to a fully connected undirected graph where each player is a node. Here the protocol attempts to establish secret keys between each and every pair of players. In this work it is assumed that there are no pre-existing secure channels which can be used to share keys. It is also assumed that every player in the network is honest.

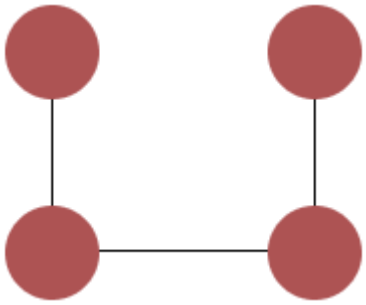
**Basic Distributed Protocol**

The most naive idea is to use the standard Merkel's puzzle between every pair of players. In this protocol, considering the presence of M honest players in the network, the query

complexity for each player is O(M*$\sqrt{N}$) because each player has to query the oracle $(M-1) * \sqrt{N}$ times and the communication complexity is also O(M*$\sqrt{N}$) because each player is sending $\sqrt{N}$ values to all (M-1) players.



But it is sufficient to make a connected graph with secure links (edges) and then the keys can be shared through these links. So here each secure link has an existing key shared between 2 players which it connects. Now if 2 players $p_i \; and \; p_j$ who are not directly connected via a secure link, then they can exchange their secret key $k_{ij}$ through edges that connect $p_i \; and \; p_j$ in the graph. So if the key is being transferred through an edge $(p_a, p_b)$, then $p_a$ will first decrypt $k_{ij}$ with shared key from previous secure link, encrypt $k_{ij}$ with $k_{ab}$ (as secure links already have keys shared through Merkle's Puzzles) and then pass it to $p_b$. Then $p_b$ will follow the same steps as $p_a$ and pass the key to the next secure link.



The disadvantage of the basic protocol is that secret key $k_{ij}$ is known to the intermediate nodes who pass the key from $p_i \; to \; p_j$. For this problem, the secret key is split and then sent in parts through multiple paths.

## Current Protocol - Details:

Instead of fixing secure links as we saw in the previous protocol, in this protocol, we arbitrarily create the secure links.

**Steps to create secure links arbitrarily:**

1. There are total N (=M*$T^2$) entries in the Hash function where M is the total number of players and T is the number of queries each player asked to the oracle.
2. Now every player asks T queries to the oracle H and obtains $T\ pairs\ (x_i,\ H(x_i))$ and stores them.
3. Given this value of N, Using the Birthday Paradox, Ω(1) secure links will be created per player because T queries of a player intersect with (M-1)T queries of other players and the total hash table size in M*$T^2$. Based on these intersections of $(x_i,\ H(x_i))$, links will be created.
4. This way a total of Ω(M) secure links will be created arbitrarily and we get a connected graph with high probability.
5. Also, The security parameter N increases from $T^2$ to M*$T^2$.

**Challenge to create secure links:**
How to make P1 and P2 realize that their Hash value matches and a secure link can be created.

**Solution1:**
Every player sends all its T entries to all the other players. This makes the communication complexity O(M*T) for every player which is very bad.

**Solution2:**
Every player sends all its T entries to a single player, say p1. This makes the communication complexity O(M*T) for p1. Can this be improved?

**Solution3 (The solution actually used):**
Every player sends its hash value (H(x)) to the player (H(x) mod M). If this receiving player detects a match in the received entries, then it informs both the other sending players and this creates a secure link between them.
As H(x) is assumed to be uniform, with high probability every player receives O(T) hash values. This makes the communication complexity O(T) for every player.

Now the keys with all the other players can be created using the idea of Basic Protocol mentioned above.

Till now for players query complexity is O(T) and communication complexity is O(T).
For adversary, query complexity is O(M*$T^2$).

As there are M secure channels created, Adversary hash values can match with any of the values from these secure channels. Hence we get a security of O(N/M) = O($T^2$) which is the same as the original Merkle's puzzle's security.

In the current protocol, solution 3 is used.

The above protocol's aim is to generate a secure connected graph. Now we will look at some arguments presented in the paper and proofs based on which we can say that it meets some requirements towards generating a secure connected graph.

**Definition: (Expander Graphs)**
Let G(V,E) be an undirected graph and $\delta > 0$, The Graph is a $\delta$-expander graph if $|N_G(U)| \geq \delta$ where U ⊂ V and $|U| \leq n/2$. Here $N_G(U)$ is the set of all vertices directly connected to set U and not in the set U. The Graph formed by setup protocol is an expander graph because it is a connected graph and satisfies the above property.

**Proof: Diameter of an expander graph with n-vertices is bounded by O(log(n))**
Let $B_t(v)$ be the set of vertices within distance t from the vertex v, then
$|B_t(v)| \geq min(n/2, (1+\delta)^t)$. We can prove this using induction.
For $t = log_{1+\delta}(n/2)$ and u,v $\varepsilon V$, we have $B_t(v) \geq n/2$ and $B_t(u) \geq n/2$. Thus $B_t(v)$ and $B_t(u)$ intersect each other. This means maximum distance between u and v is $log(n)$. Hence the proof.

**Setup protocol Analysis:**

Correctness: For correctness that the keys exchanged using above protocol between every pair of players $p_i$ and $p_j$, $k_{ij}$ and $k_{ji}$ must be equal. Since the values of $k_{ij}$ and $k_{ji}$ will be the preimage of the collided hash values between $p_i$ and $p_j$ , the probability of correctness is based on the probability of collision in the Hash table. Based on this the probability that for all pairs keys are exchanged correctly is shown to be $\geq 1 - T^{-2}$.

**Query and Communication Complexity:**

For the initial distribution of messages to establish enough data for detecting collisions, first it is proved in the paper that for the above protocol each player communicates at most $8T$ messages except at most $M.2^{-T}$ probability. The communication involves both sending and receiving messages, so a player $p_i$ will be sending at most T messages of the form $(x , H(x))$ in order to get matched. For receiving, the player will be receiving messages from other players $p_j$ where $H(x)\%M = i$. Using this probabilistic analysis the bound of $M.2^{-T}$ was proved, which was negligible for reasonable value of T and thus each player can be assumed to communicate at most $8T$ messages with high probability.

After this, the communication complexity involved in sending and receiving messages about detected collisions is calculated. Since for each collision message is sent to 2 players whose hash values collide with each other, the number of messages is upper bound by twice the

number of “good” (assuming correctness with high probability) collisions. So in the paper first the probability of number of collisions lower bound by $65\ log\ T \cdot T$ is estimated, it is proved to be less than $(36\ log\ T \cdot T\ )^{-1} + T^{-2}$ which is small for a reasonable value of T. So the number of messages in this case can also be assumed to be at most $2\ \times\ 65\ log\ T \cdot T\ =\ 130\ log(T)\ T.$

So from the previous 2 results it was concluded that the amount of communication was bound by $O(T\ )\ \ except\ with\ probability\ at\ most\ M.\ 2^{-T}\ +\ (36\ log\ T \cdot T\ )\ -\ 1\ +\ T^{-2}$), or simply it is $O(T\ )$ with high probability.

**Connectivity:**

It is proved in the paper that the graph generated by the setup protocol is an expander graph with high probability.

1. First the definition of a useful group was defined, in which a group of player’s U was said to be useful if the group made at least $kT/2$ distinct queries to the oracle, where k is the number of nodes in U. Then it was proved that any group U of the players was useful with high probability, (except with probability at most $2^{-2T}$).

2. Using the above result it was then proved that the neighborhood of any group U in the graph generated by the setup protocol was of size $\geq\ k/2,\ where\ k\ is\ the\ number\ of\ players\ (nodes)$ in the group with high probability. For this proof, two scenarios were used, that is if the group U is either useful or not. If the group was useful and if the size of its neighborhood was less than $k/2$, then using the definition of useful group it was shown that probability of elements not in the neighborhood of U and not colliding with the queries of U was very less ($e^{-R\cdot k/8}$) and thus after taking union bound $Pr[|size\ of\ neighborhood\ of\ (U\ )| \leq k/2\ |\ U\ is\ useful]\ \leq e^{-R\cdot k/12}$. And from the previous paragraph we can say that $Pr(U\ not\ useful)\ <=\ 2^{-2T}$. Adding both the above cases we can say that with high probability that size of neighborhood of any group is greater than $where\ k\ is\ the\ number\ of\ players\ (nodes).$

Finally using 1 and 2, the paper proves that with high probability, the graph obtained is a $\delta$ expander and thus diameter is bound by $4.\ log(M)$ with high probability.

**Security:**

The adversary has the full transcript of the protocol and if he makes at most $T_A$ queries to H then one of his queries can be any of the keys of the protocol with probability at most $T_A/N$.

Here the security of the setup protocol is not strong enough (quadratic gap, same as standard Merkle puzzle). As a result the amplification protocol was introduced as follows.

## Amplification

A single round of the setup protocol mentioned above does not significantly improve security over the basic protocol. Hence a small parameter L is introduced. The setup protocol is executed L times with domain separated hash functions now and during each execution only a part of secret key (say $k_{ij}$) is sent on any path to $p_j$ using the secret keys of intermediate players. Due to this the adversary will now have to retrieve keys from each of the L executions to get the original key.

Let $k_a$ be the part part in $a^{th}$ execution, then

$K_{ij} = k_1 \oplus k_2 \dots \oplus k_l$

Also since it is proved in the paper that the diameter of graph obtained through the setup protocol is bounded by $O(log(M))$, the length of the paths during every iteration will be bounded by $O(log(M))$ and thus the adversary will also have a limited number of links for attack.

**Amplification protocol Analysis:**

**Security Analysis:**

It is shown in the paper that after making around N/4 queries to the oracle and the encryption function in the amplification protocol setup and key distribution, the information about the key $k_{ij}$ between players $p_i \; and \; p_j$ still looks uniformly random in [N], except a small probability of $T^{-6}$ to the adversary. Now in that proof it can be seen that adding multiple iterations of the setup protocol lead to strengthening of security as now every chunk of key across the L iterations of the setup protocol now need to be in the set of queries which were made by the adversary.

### Optimality of the Distributed Key Agreement Protocol

Let $\Pi$ be a protocol between M players using a random oracle H in which:

1. Every player make at most T queries to H
2. Player1 (p1) outputs j $\varepsilon$ [M]\{1} and $k_{1,j}$ and player(j) outputs $kj, 1$ such that Pr[$k_{1,j} = k_{j,1}$] $\geq \alpha$, then for every $0 < \delta < \alpha$, there is an adversary that makes 400.M.$T^2/\delta^2$ queries to H and outputs $kj, 1$ with probability $\alpha - \delta$.

**Proof:**

We can consider this as a two players problem where one player is p1 and other player is { $p_1, p_2, \dots, p_n$} and all the messages intended for the $2^{nd}$ set is going to the second player.

Finally, P1 outputs $k_{1,j}$ and the second player outputs $kj, 1$. Using the standard merkel's puzzle for 2 players, the above statement holds true.

### Implementation:

Our implementation of the distributed merkle's puzzle can be found here: https://colab.research.google.com/drive/1UnhwzP49fmvgY15UKQvDgWI6h_g4VvNt?usp=sharing

In this implementation, we have shown the setup protocol which forms the sparse connected graph. Then using that graph, we have constructed a fully connected graph. Finally, using the amplification method, we have enhanced the security of the model.

### Implementation Result:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 150403072 | 245376558 | 128118729 | 243267543 | 163788592 | 172157976 | 66614995 | 100801414 | 133761190 | ... |
| **1** | 150403072 | 0 | 53432668 | 116326243 | 185310782 | 77923820 | 150901789 | 90719178 | 242967473 | 102944588 | ... |
| **2** | 245376558 | 53432668 | 0 | 72730570 | 63933535 | 100989313 | 47849474 | 146521057 | 165846332 | 207852446 | ... |
| **3** | 128118729 | 116326243 | 72730570 | 0 | 114920442 | 234481245 | 215816772 | 97112282 | 126586781 | 49735821 | ... |
| **4** | 243267543 | 185310782 | 63933535 | 114920442 | 0 | 81073110 | 113471096 | 123806824 | 82239227 | 230635581 | ... |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **59** | 92079262 | 76891853 | 103026485 | 238585532 | 14379655 | 174163268 | 90045747 | 23654192 | 26425413 | 223185242 | ... |
| **60** | 27407467 | 6656297 | 125067975 | 230127774 | 138600142 | 84129953 | 73386970 | 241334507 | 77843499 | 97468998 | ... |
| **61** | 10072611 | 97088501 | 224142436 | 245199101 | 90687891 | 106552236 | 195495230 | 174290918 | 103335887 | 193182225 | ... |
| **62** | 178625002 | 207218530 | 168291015 | 180970684 | 241791577 | 96440441 | 37340969 | 153557780 | 212510972 | 189512211 | ... |
| **63** | 122213496 | 138726285 | 95566141 | 164369958 | 127839523 | 135116078 | 88681412 | 242483216 | 20147749 | 104059252 | ... |

64 rows × 64 columns

These are the keys setup by the distributed merkle's puzzle implementation between each pair of players..

### Results:

Using this, the adversary can't retrieve the key directly and he needs all the L parts of the key to retrieve the entire key. Hence the adversary needs the entire hash table with $O(M*T^2)$ entries to get any complete key. So we get a security of $O(M*T^2)$.
Security/Work ratio is O(M*T) which is the improvement over the Security/Work ratio of O(T) of Original Merkle's Puzzle.

### Possible Extensions:

This paper assumes that all the players in this distributed setting are honest but that is not the case. We are considering some persons who are semi-honest (they are trying to learn about the keys of other persons as the part of the key passes through them).

If the semi-honest person is not present in each path (as discussed in the amplification section), then he will not be able to extract the key. We need to increase the probability that such scenarios don't occur. Some possible solutions are:

1. We can fix some paths along the honest players(assuming we know such players), this will reduce the possibility of honest-player being in the path because the path length is upper bounded by $O(log(M))$.
2. We can increase the value of L(as discussed in the amplification section), this will lower the probability but it will also increase the query and computational complexity.

## Challenges

1. This algorithm only gives quadratic security in comparison to other algorithms which gives much higher degree polynomial security.
2. Identification of semi-honest players in a distributed setting is always a big challenge as even one key extraction by him can break the system.