Ontropy: Virtual Rollup Protocol

Alexander Atamanov and José Betancourt

Ontropy Inc., USA

28.07.2023

The Ontropy Virtual Rollup protocol has been designed to increase the amount of compute and data that can be trustlessy managed while drastically minimizing user friction across a broad spectrum of Web3 applications. The Virtual Rollup Protocol solves many foundational UX problems common for Web3 applications such as latency, gas fees, multiple wallet transaction clicks, and more. For developers, Virtual Rollups are easy to integrate as dataflows remain unchanged with only additional cryptographic signatures added to outgoing data and reaggregation to incoming data. Additionally, Virtual Rollups are chain agnostic. This paper explores Ontropy's solution and our multi-faceted approach to rethinking Web3 UX and throughput.

Alexander Atamanov: alexander@ontropy.io, https://ontropy.io

José Betancourt: jose@ontropy.io, https://ontropy.io

1 Introduction

The concept of cryptocurrencies, first described by Satoshi Nakamoto, employed a vision of purely peer-to-peer electronic cash. Elaborating on that idea, Vitalik Buterin invented Ethereum after his character in "World of Warcraft" computer game was cruelly nerfed by the developers. Ethereum gave the world the first practical implementation of a smart contract concept. Yet, 14 years after the Bitcoin release and seven years after the Ethereum release, both original promises are not fully fulfilled. You can't exchange cryptocurrency anonymously and peer-to-peer, and the games are still centralized. There are some Web3 games, but the value added by the presence of some of the game elements on the chain is doubted by many and very few of them are truly permissionless or decentralized.

Why are games important? Because games often served as an early adoption medium for many computer technologies. Why is Web3 adoption lagging? Because you have to pay for the transactions - and often a significant amount - and the UX is bad. Value added is doubted by many.

Many Web3 applications, such as gaming platforms, trading systems, swaps, and DEXes, often involve complex multi-step operations that require significant user interaction and result in high transaction costs. These interactions and costs can hinder the usability and adoption of such applications, limiting their scalability and effectiveness.

Layer 2 scaling solutions were proposed and built to solve the issue of scalability and improve user experience in blockchain applications. They function by processing the majority of transactions off the main Ethereum chain (Layer 1), thereby significantly reducing costs and increasing transaction speeds. Layer 2 solutions such as Optimistic Rollups and ZK Rollups have shown promise in dealing with these issues.

These Layer 2 protocols work by bundling or 'rolling up' multiple transactions into a single one, which is then committed to the main chain. While this increases efficiency, it is not without its problems. For instance, Optimistic Rollups require a long challenge period, during which fraudulent transactions can be disputed, leading to delays. ZK Rollups, on the other hand, require complex zero-knowledge proof constructions that can be computationally intensive and challenging to implement.

And more to that, the fundamental UX problem - that the user is required to interrupt the normal session flow and perform an on-chain transaction even for actions that nobody disputes, remains intact even with the L2 Solutions. Optimistic rollups do not get the job done as they still to require the transaction.

StarkNet and DyDx provide excellent examples of different players who have begun to address those issues. StarkNet uses zk-STARKs to create a scalability solution that can handle a higher number of transactions, while DyDx uses a hybrid solution that merges off-chain order matching with on-chain settlement. These solutions successfully circumvent the congestion of the Ethereum mainnet and reduce gas fees, but they compromise on the essential tenets of decentralization and peer-to-peer interactions.

Take StarkNet for instance, although it enhances scalability and reduces transaction fees, it also introduces a degree of centralization. StarkNet's zk-STARK proofs, although powerful, require a specialized, centralized prover environment for their generation. This centralized aspect of StarkNet is in contrast to the ethos of complete decentralization that Ethereum and its applications aim to uphold.

Similarly, DyDx, while successfully merging off-chain order matching with on-chain settlement, isn't a true peer-to-peer platform. Users are reliant on the platform's order book for trade matching instead of directly interacting with their peers. This reliance means users are partially dependent on the platform for their interactions, reducing the autonomy they might have on a fully decentralized, peer-to-peer platform.

In stark contrast, Ontropy Virtual Rollup ensures that all computations are done peer-to-peer between the end-users without introducing centralization. It maintains a strong commitment to decentralization and peer-to-peer interactions while addressing the limitations of gas fees and latency. It is through this balanced approach that Ontropy Virtual Rollup aims to provide a pathway for more mainstream adoption of Ethereum-based applications, including Web3 games.

As a side effect, Ontropy Virtual Rollup basically allows protocols to reduce server costs by transferring a lot of computation to the often underutilized endpoints. That is achieved by the multi-party computation protocols baked deep into the Ontropy SDK.

The current landscape of Web3 applications lacks an efficient and secure protocol that minimizes user interaction and transaction costs across a broad spectrum of multi-step operations. Existing solutions often suffer from scalability issues, privacy concerns, and a lack of seamless integration with different applications.

Moreover, in the context of multi-player, multi-round games, where frequent and numerous transactions are the norm, the absence of a streamlined protocol exacerbates the challenges faced by both developers and users. The need for an optimized solution that reduces transaction costs enhances privacy, and ensures trust and integrity within sessions is evident.

Therefore, there is a clear demand for a cutting-edge protocol that addresses these challenges and provides a seamless, cost-effective, and secure framework for executing multi-step operations in Web3 applications. The protocol should not only minimize user interaction but also incorporate privacy-preserving mechanisms and have efficient and fast dispute resolution.

2 Ontropy Solution: a Helicopter View

The Ontropy Virtual Rollups protocol presents a solution to address the challenges faced by Web3 applications involving multi-step operations. By minimizing user interaction and transaction costs, the protocol offers a streamlined and efficient framework for a broad range of applications, including gaming platforms, trading systems, swaps, and DEXes. Blockchain achieves security through decentralized consensus. However, in cases with a relatively small number of participants acting in groups, consensus can be easily reached without executing the actual transaction on-chain.

The Ontropy Virtual Rollups protocol enables direct peer-to-peer agreements to minimize costs and optimize performance by providing an easy-to-use plug-and-play cryptographic solution that includes low-latency advanced peer-to-peer networking toolkit.

Virtual Rollups are presented in a form of an SDK that is available for multiple platforms including Web, Unity, iOS, Android and Python.

It provides capabilities to reach an agreement on the outcome of any process with the cryptographic algorithms that employ multi-party computation on the end-user's devices. If the agreement has been reached, then there is no need to employ full blockchain security. It will be employed when the agreement could not be reached, which in turn minimizes the chances of that actually happening (in full accordance with the Game Theory).

State channels have been a cornerstone of computation scalability strategies in blockchain environments. They offer a mechanism for participants to interact in a private, truly peer-to-peer manner, with only the final state of their interaction committed to the blockchain. This significantly reduces the computational burden on the main net and alleviates concerns over transaction speed and cost.

In traditional state channels, two parties engage in a series of transactions, keeping the majority of them peer-to-peer and only submitting the final state of their transactions to the blockchain. This method, although beneficial, has limitations in multi-party systems and lacks the flexibility for participants to join or leave the channel dynamically.

Ontropy pushes the boundary of state channel technology by introducing multi-party channels with dynamic participant management. It's not just a two-party game anymore; it can be an interactive session with multiple participants, where parties can freely join and leave as per their needs.

Moreover, Ontropy's approach employs advanced cryptographic primitives including BLS (Boneh-Lynn-Shacham) and Schnorr signatures. BLS signatures are a type of non-interactive aggregation signature that provide a short, constant size signature regardless of the number of signers, making them particularly suitable for multi-party state channels. Similarly, Schnorr signatures provide robust security properties and the ability to aggregate signatures, thereby improving the efficiency of multi-signature verifications in multi-party settings.

The application of these advanced cryptographic primitives, coupled with the dynamic and inclusive nature of Ontropy's Virtual Rollup, brings forth a solution that not only resolves the scalability concerns but also promotes an enhanced level of interaction and versatility in blockchain applications. This innovative design will transform how we understand and use Web 3 Applications, extending their reach.

Could there be a disagreement in many edge cases, espicially in the high number of uses setting? Great question! Yes, it can. And Ontropy got a way to settle disputes and disagreements by employing complete security of the underlying blockchain in a cost-effective manner by a Virtual Rollup smart contract deployed on most EVM-compatible L1 and L2. This approach balances streamlined consensus and maintains security in multistep operations.

At the core of the Ontropy solution is the concept of Virtual Rollups, which encapsulates the entire multi-step operation within a session. Participants generate new ephemeral key pair and commit to that keys on-chain. They then perform fully verified by all the participants and if total agreement could be reached (which is the case for most of the interactions) the process continues smoothly without unnecessary on-chain transactions. This approach eliminates the need for individual on-chain transactions for each step, significantly reducing the associated costs and interactions thus significantly improving efficiency. The protocol initiates the rollup process by locking the required funds in a smart contract. Upon successful initiation, participants engage in a Distributed Key Generation (DKG) process, ensuring secure and fair distribution of cryptographic keys. This step establishes trust and integrity within the session, which is vital for maintaining a secure environment. Within the session, participants can perform various operations, such as placing bets, making game moves, or executing trades. The correctness of these operations is validated using BLS Signarues, Schnorr Signatures or Zero-Knowledge Proofs (ZKPs), which are exchanged peer-to-peer and offer privacy-preserving mechanisms while ensuring the accuracy of computations. Additionally, player balances are updated based on the outcomes of these operations, keeping track of the session progress, and employing a vision of a dynamic state channel.

As such, each participant holds a complete and cryptographically verified state of the session. To enhance trust and verification, each stage of the rollup is authenticated using session multi-signatures. Participants sign the necessary requests, adding an extra layer of security and consensus to the session. When a participant decides to exit the rollup session, they submit a cash-out request, which requires the signatures of all participants to indicate consensus on the outcome. Finally, the protocol enables fund unlocking by submitting the signed cash-out request to the smart contract, leading to the secure transfer of funds back to the user's account.

The Ontropy Virtual Rollups protocol revolutionizes multi- step operations in Web3 applications by providing a seamless, cost-effective, and secure solution without compromising on security. It significantly reduces transaction costs and interactions, making it particularly advantageous for applications involving frequent and numerous transactions. By integrating features such as DKG, MPC, ZKPS, session signatures, and consensus-driven cash-out requests, the Ontropy Virtual Rollup ensures efficient and secure execution, catering to the evolving needs of the Web3 ecosystem.

3 Generalised Virtual Rollup Protocol

As a general solution, we propose an algorithm where most of the interaction flow is conducted peer-to-peer, and each step is verified by a Schnorr or BLS signature. For a list of critical actions Merkle-tree proofs are stored, and those events are propagated through all of the peer-to-peer network so that all the participants have got the same view on the critical events. Those events include players switching lobbies, dropouts, and others. The number of that events should be limited though.

Schnorr signatures and Merkle proofs have been selected as they could be verified on-chain very effectively in terms of gas consumption. If the participant did drop out, he has got a timeout period τ in which he can emit his actions on-chain via the smart contract. All players are required to listen to the blockchain and include the emitted event in any game step.

In case of collusion or disagreement, either the signature or the proof of the game history would be needed, and the collision resolution will happen on-chain for that session round by the smart contract. The party that has lost the dispute will be penalized. The penalty could be set by configurable session rules.

The steps do include the previously described processes in Section 3.

3.1 General Protocol Description

This protocol provides a description of Ontropy Virtual Rollup using Schnorr signatures-based system for playing a random game on a use case of a multi-table poker tournament. The protocol aims to enhance the security, efficiency, and fairness of the game by leveraging innovative cryptographic techniques. This document uses poker and other random games as an example.

3.1.1 Random Game and Multi-Table Poker Tournament

The protocol supports a random game with a number of players playing in smaller groups - like a thousand players playing in a poker tournament split by tables with 9 participants. These game types have been chosen due to their popularity and the complexity they introduce, allowing us to demonstrate the effectiveness of the protocol in various scenarios. It can easily be extended beyond gaming reality.

3.1.2 Key Management

Participants generate an ephemeral key pair upon joining the session (game). The public key is committed to the smart contract, and the private key is used for signing actions during the game. An argument for knowledge of the private key is provided to mitigate rogue-key-like attacks. This approach offers increased security and enables more actions to be completed off-chain. However, it may result in higher initial transaction costs and complexity.

3.1.3 Random Game Mechanics

a. Game Initialization

When a session (game) is initialized, players deposit a predetermined amount of funds into the smart contract as their buy-in. The smart contract generates a unique game ID and assigns each player a unique player ID. Public keys of all players are emitted from the smart contract upon game initialization and when the player list changes, ensuring transparency and providing necessary information to all participants.

b. Randomness Generation

Randomness in the game is achieved using modified Pedersen commitment scheme (as stated in the next chapter). Pedersen commitments utilize a commitment hash and pre-commitment to the Pedersen value to prevent the last actor from biasing the RNG by withholding their commitment until they see the commitments of all other participants. This ensures a fair and unbiased generation of random values for the game.

c. Schnorr Signatures and Off-chain Computations

Players use Schnorr signatures to sign their actions in the game, such as betting, folding, or raising. The recursive signature technique or Merkle Proofs are employed to verify each game state, creating a chain of trust where each signature implicitly verifies all the previous ones. This approach improves efficiency and allows for off-chain computations, reducing the on-chain transaction costs. In case of a dispute on signature validity, all players are required to provide the current game state to the smart contract for verification. Any misbehaving player will be excluded from the game, ensuring the integrity of the gameplay.

d. On-chain Verification

On-chain verification is crucial for maintaining the fairness and transparency of the game's outcome. Schnorr signature verification is performed on-chain using the ECRECOVER precompile, which enables AA wallets or other smart contracts to verify Schnorr signatures at a gas cost similar to regular ECDSA signatures (6000 gas). This affordable on-chain verification ensures the accessibility of the game to a wide range of participants.

e. Handling Player Drop-outs

If a player drops out of the game, other players can attest to that fact. The dropped player has N blocks (configurable) to claim their reconnection by calling a smart contract function that emits their UUID. If the dropped player fails to claim reconnection within the specified timeframe, players proceed with the game without the dropped player. In such cases, the remaining stages are completed, and the final game results are posted on-chain, ensuring the completion of the game.

3.1.4 Multi-Table Poker Tournament

a. Tournament Initialization

The multi-table poker tournament starts with a predetermined number of tables and players. Each player deposits their buy-in into the smart contract, which assigns them a unique player ID and randomly assigns them to a table.

b. Tournament Progression and Table Transitions

As the tournament progresses, players may be eliminated or required to move to different tables based on their performance. The smart contract updates its table assignment and shares the player's current balance and past game history with the new table.

c. Player Verification at the New Table

When a player joins a new table, other players at the table can verify the incoming player's balance and past game history using the information provided by the smart contract. This ensures the accuracy of the new player's balance and detects any cheating or suspicious behavior in previous games.

d. Final Table and Prize Distribution

The tournament continues until a final table is reached, consisting of the top players from the tournament. The final table determines the championship and the distribution of the prize pool. On-chain verification using the ecrecover precompile confirms the tournament's outcome, ensuring fairness and transparency. The smart contract then distributes the prize pool to the winners based on their final standings.

3.1.5 System Architecture Flow Diagram

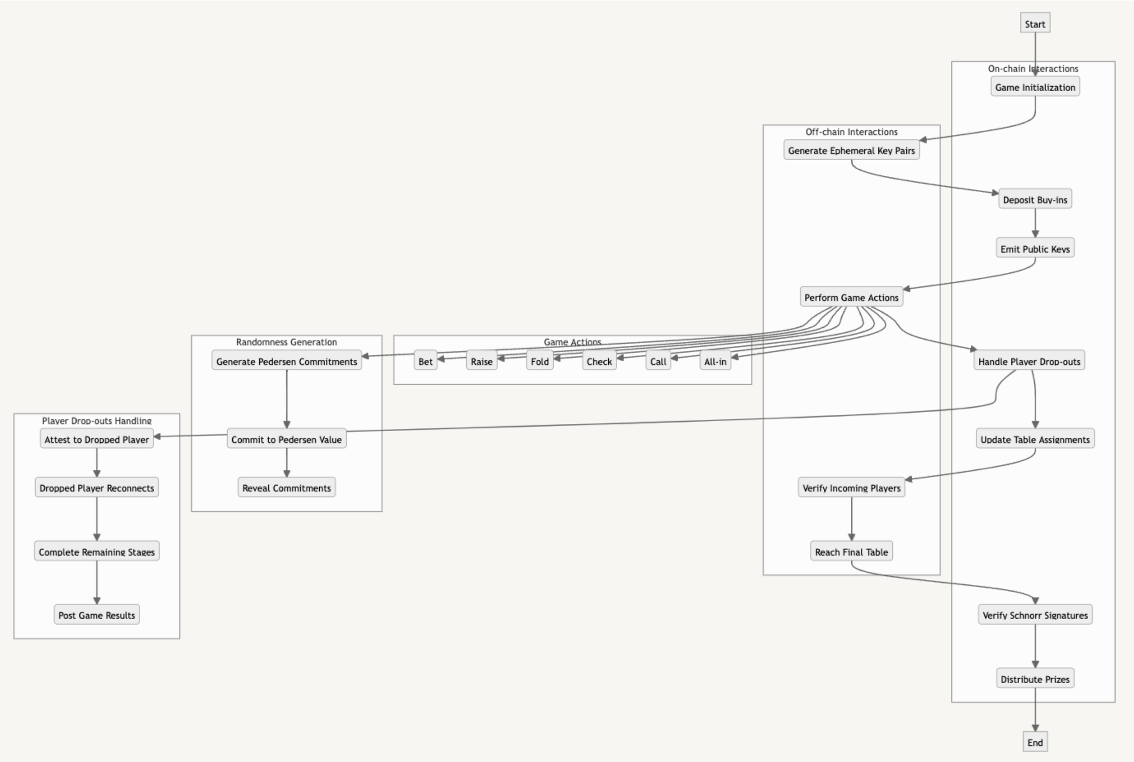

Figure 1: System Architecture Diagram

3.1.6 Questions and Considerations

1. Should players wait for a certain number of confirmations to continue playing?

Players should wait for their buy-ins to be confirmed and for the smart contract to generate and emit their public keys before they can start playing.

2. Is off-chain play possible after a player has reconnected?

Yes, off-chain play is possible after a player has reconnected. The dropped player has N blocks to claim they are back online on-chain by calling a smart contract function that emits their UUID. After reconnection, the player can continue participating in the game using off-chain computations and signing actions with Schnorr signatures.

3. A detailed algorithm for hand-to-hand play on the payout change is needed.

The following steps can be implemented:

a. Players sign their actions (betting, folding, raising) using Schnorr signatures. b. Perform off-chain computations to calculate the pot size, determine the winner, and distribute the winnings. c. Use Pedersen commitments for randomness generation in the game. d. In case of a dispute, provide the current game state to the smart contract for on-chain verification. e. Update the player's balance and game history based on the outcome of each hand.

4. Consider implementing the correctness of computation proofs for game logic validation.

To implement the correctness of computation proofs for game logic validation, the protocol can use zero-knowledge proofs or zk-SNARKs. These cryptographic techniques allow players to prove that they have performed the correct computations without revealing any sensitive information about their actions or the game state.

5. How can the system handle network latency and potential synchronization issues between players?

The system can handle network latency and potential synchronization issues by using a combination of on-chain and off-chain interactions. Off-chain computations and signing actions with Schnorr signatures can help reduce latency and synchronization issues. In case of disputes or inconsistencies, players can provide the current game state to the smart contract for on-chain verification.

6. What mechanisms can be put in place to prevent players from colluding or cheating during the session?

To prevent participant from colluding or cheating during the game, the protocol can implement the following mechanisms:

a. Use Pedersen commitments for randomness generation to prevent players from biasing the RNG. b. Require players to provide an argument of knowledge of their private keys to mitigate rogue-key-like attacks. c. Perform on-chain verification of Schnorr signatures to ensure the fairness and transparency of the game's outcome. d. Verify incoming players' balances and game history when they join a new table in a multi-table poker tournament.

4 Ontropy Virtual Rollup Solution for Mental Poker

In this section, we describe the current state of Ontropy Virtual Rollup development in regard to the Mental Poker problem. We also discuss practical aspects and perspective solutions we are considering.

4.1 Mental Poker Solution

4.1.1 Distributed Key Generation

For the distributed key generation, we use a modified version the method shown by Pedersen. It goes as follows:

1. Every player i generates a random polynomial $f_i(x) = b_{i,0} + b_{i,1}x + \ldots + b_{i,k}x^k$ where $k = \#$ number of players -1 and $b_{i,0} = s_i$.

2. Then, every player i computes $B_{i,j} = g^{b_{i,j}}$ for $j \in 0, \ldots, k$ as well as $f_i(j)$ mod$q$ for $j \in \{0, \ldots, k\}$.

3. $B_{i,j}$ and $f_i(j)$ are then sent to players $j \in \{0, \ldots, k\}$.

4. Each player j verifies $g^{f_i(j)} = \prod_{m=0}^{k} (B_{i,m})^{j^m} \bmod p$. Failure to verify for values sent by user i means they're cheating, and a new instance of the protocol can be established without them.

5. After verification, the public key $y = \prod g^{s_i} \bmod p$ for all players i.

6. The value s_i is player i 's private key were the full private key is $x = \sum s_i$ mod$q$, although that is never assembled in the protocol.

4.1.2 Generating Randomness

To generate the distributed randomness necessary we first define $\mathbb{N}_D$ which is the set of values our randomness can be in (card games - $D = 52$, roulette $D = 36$, coin toss $-D = 2$, etc.). We then proceed as follows:

1. Each player i chooses their random seed $c_i \in \mathbb{N}_D$

2. They compute the ciphertext $E(g^{c_i})$ and send a cryptographic commitment $C(E(g^{c_i}))$ to all other players.

3. After the commitments have been verified, the ciphertexts are revealed and players compute $\prod E(g^{c_i}) = E\left(g^R\right)$ where $R = \sum c_i \bmod D$

Every player now has access to the encrypted version of the final randomness value R.

4.1.3 Revealing Public Randomness

In the case where R is meant to be a publicly accessed value such as a faceup card or a roulette spin result, the players all reveal their seeds c_i and their randomness coefficients e_i used in computing $E(g^{c_i})$. Players can then verify c_i is in the correct domain (although if it is not, it is trivial to show that cheating of this sort is relatively harmless) and that the ciphertexts provided are in fact correct. If all the encryptions are valid, the players can compute $R = \sum c_i$ mod$D$. If any of the verification steps fail, the fair players run a new DKG and establish a new group. The punishment for the cheating player depends on where this protocol is implemented and is outside of the scope of this paper. For games that don't require collision detection and private randomness such as coin toss or roulette, the protocol as described up to this point is sufficient. The following sections deal with matters regarding collision detection and private random generation (e.g. face-down cards).

4.1.4 Generating Private Randomness

The protocol also allows us to create secret randomness values that only a limited subset of players know. For the sake of simplicity, we will show the case of one player j but it is trivial to adapt this part of the protocol for a proper subset of players with a cardinality larger than one. Essentially, players follow the steps from the previous section but instead of revealing their seeds c_i to all, they only send them to player j. He then computes $R = \sum c_i \bmod D$ and needs to prove that result and commit to it. To do this player j has to:

1. Prove $E\left(g^R\right)$ is an encryption of a value in the set $\left\{g^0, \ldots, g^D\right\}$ using the encryption subset composition proof.

2. Prove that $\frac{E\left(g^{\sum c_i}\right)}{E(g^R)}$ encrypts a value in $\left\{g^0, g^D, \ldots, g^{D(k-1)}\right\}$ where $k =$ #number of players.

These steps are sufficient because if player j decides to alter the value of R, $\frac{E\left(g^{\sum \epsilon_i}\right)}{E(g^R)}$ will not provably belong in $\left\{g^0, g^D, \ldots, g^{D(k-1)}\right\}$.

4.1.5 Encryption Subset Composition Proof

This section describes a proof of encryption subset membership based on a modified version of the Disjunctive Schnorr Signature[5]. We apply the following algorithm to generate the values S_i :

Then, for each set $S_i = \{g^0, g^{\gamma_i}\}$ the prover performs the following recursive procedure while $\Omega_i > 0$:

1. Set $\Omega_0 = R$
2. If $\Omega_i < \gamma_i$, we take $\chi_i = (1, g^0)$
3. If $\Omega_i \geq \gamma_i$, we take $\chi_i = (1, g^{\gamma_i})$
4. $\Omega_{i+1} = \Omega_i - \gamma_i$

At the end we should have $R = \sum \gamma_i$. For each set S_i, the prover takes its corresponding ciphertext $\chi_i = (1, g^\sigma)$ where σ is either 0 or γ_i and computes $E\left(g^\sigma\right) = (\alpha, \beta) = \Theta_i$. They then make the following fie

0 $(Y_1, G_1) = \left(\alpha, \frac{\beta}{g^0}\right)$

- $(Y_2, G_2) = \left(\alpha, \frac{\beta}{g^{\gamma_i}}\right)$

Let's assume w.l.o.g. assume $\gamma = g^0$ and (Y_1, G_1) is the valid transformed pair of keys:

1. The player shares (Y_1, G_1) and (Y_2, G_2).
2. Everyone calculates $w_i = G_1^{\lambda_i}$ where λ_i is randomly chosen $\in \mathbb{Z}_q^*$.
3. The prover calculates $w_2 = G_2^{d_2} \cdot Y_2^{-c_2}$ where d_2 and c_2 are random values $\in \mathbb{Z}_q^*$
4. All players agree on $c_1 = \text{hash}\left(G_1, Y_1, G_2, Y_2\right)$ where hash() is an arbitrary hash function.

5. Everyone computes $d_i = \lambda_i + c_1 \cdot s_i \bmod q$ where s_i is their piece of the private key.

6. Everyone sends d_i and w_i to the prover who can then calculates $d_1 = \sum d_i$ and $w_1 = \prod w_i$ 7. The prover reveals $d_1, w_1, w_2, c_1, c_2, d_2$ and the other players can check if $G_i^{d_i} = w_i \cdot Y_i^{c_i} \bmod p$ for $i \in \{1, 2\}$

This procedure proves that each ciphertext Θ_i is a valid encryption of a value in the set S_i without revealing any information about the value itself. After all the Θ_i 's are signed and verified, all players compute and decrypt $\frac{\Pi\Theta_i}{E(g^H)} = (\alpha, \beta)$ and verify that $\frac{\beta}{\prod \alpha^{s_i}} = 1$ where α^{s_i} is a piece shared by each player. Thus, we can show that $E\left(g^R\right)$ encrypts a value in the set $\left\{g^0, \ldots, g^D\right\}$ without revealing it.

4.2 Modified Encryption Subset Composition Proof

The procedure is identical to the one described in the previous section but we have k sets of $\left\{g^0, g^D\right\}$. This is used to prove that $\frac{E\left(g^{\sum c_i}\right)}{E(g^R)}$ is an encryption of a value in the set $\left\{g^0, g^D, \ldots, g^{D\cdot(k-1)}\right\}$.

4.3 Detecting Repetitive Random Values

In some cases where D is a small number, the same random value can be generated multiple times which is undesirable for games such as poker, blackjack, some RPG games, etc. This leads to complications and we need a mechanism for detecting such collisions. To do so, all players will keep a record of all previous encryptions $E\left(g^{R'}\right)$ in a set L. After a new random encryption $E\left(g^R\right)$ is generated, we can compare it with all previous values in the set L without revealing any information about R. Let's take $E\left(g^R\right)$ which is the randomness to be checked and $E\left(g^{R'}\right) \in L$. Players can use the multiplicative homomorphism of the EC ElGamal encryption and Schnorr Signatures to detect repeating values in the following way:

1. If $E\left(g^R\right) = (a_1, b_1)$ and $E\left(g^{R'}\right) = (a_2, b_2)$, players compute $\frac{E(g^r)}{E\left(g^{R'}\right)} = \left(\frac{a_1}{a_2}, \frac{b_1}{b_2}\right) = (A, B)$.

2. Participants do the following transformations for $\lambda_i = \text{hash}(g, y, A, B, j)$ where $j \in \{1, 2\}$ and hash() is an arbitrary hash function:

$$G = g^{\lambda_1} \cdot A^{\lambda_2} \bmod p$$
$$Y = y^{\lambda_1} \cdot B^{\lambda_2} \bmod p$$

3. Individual players choose a random $k_i \in \mathbb{Z}_q^*$ and compute $w_i = G^{k_i} \bmod p$ and $l_i = g^{k_i} \bmod p$.

4. Players share w_i and l_i and compute and agree on $w = \prod w_i \bmod p$.

5. Players then compute $c = hash(w)$ and $t_i = k_i - c \cdot s_i \bmod q$ where s_i is their piece of the private key. 6. All players share their t_i and others verify that $g^{t_i} = \frac{l_i}{\left(g^{g_i}\right)^c} \bmod p$ for each player.

7. If all verification steps are correct, players compute and agree on the value of $t = \sum t_i \bmod q$.

8. Players check if $w = Y^c \cdot G^t \bmod p$. If so, a collision has occurred and the randomness should be recomputed. If the equation doesn't hold, the value is unique and is ready to be used and saved to the set L.

4.4 Process

1. Users engage peer-to-peer to obtain the data via Dynamic User Entropy or through the needed distributed keys mentioned thereof. Each user stores the desired, underlying data, c_i, and the signed proof, $C\left(E\left(c_i\right)\right.$, from every other user within the scheme. Consensus on the data, randomness, price feed, or otherwise has been reached.

2. One or many users will now commit the collectively agreed data, c_i, along with a stake s to chain for use within these users, exclusively. c_i will be trusted by default and used for the transaction as all parties will have the chance to dispute the committed data by attaching the collected proof, $C\left(E\left(c_i\right)\right.$, along with a stake s.

3. The user who submitted c_i will now submit their $C\left(E\left(c_i\right)\right.$ or forfeit the dispute. Because $C\left(E\left(c_i\right)\right.$ represents an individual agreement from the adversarial party using the distributed key to the validity of c_i both proofs cannot be simultaneously true.

4. At this stage, the system and userbase are alerted to the presence of a dispute and any willing users may cast votes in either direction by staking some set amount s.

5. After a short set time, by simple majority of users, the true result is determined, becoming or remaining the value used. The losers' staked capital is forfeited and distributed to the winners.

6. Because a short voting time frame must be used for efficiency, it is possible a group of malicious users could win a majority of the round if honest participants have low turnout (although this is unlikely as a verifiably false result, especially one with low honest stakers, would mean a high reward for the honest winners). To deter this, any user in the system can now challenge the result again by staking a higher amount $100 \cdot s$ to the losing side.

5 Modern Cryptographic Primitives Made Accessible

Ontropy not only propose an advanced scaling solution, that is simultaneously effective and elegant, but also provides the toolkit to make it work.

The foundational blocks of Ontropy Virtual Rollups are networking and cryptography. The networking component is based on libp2p. It's the library used by Ethereum, PolkaDot and many other great protocols. But in those protocols, it is used to connect nodes. In Ontropy it's used to connect clients themselves.

Libp2p is a modular network stack that allows you to build your own peer-to-peer applications. It provides the essential building blocks for a network and enables applications to be device, bandwidth, latency, and connectivity agnostic.

One of the key models libp2p uses is the publish-subscribe (pub-sub) model. This is a message communication pattern where senders (publishers) categorize published messages into topics, and other network peers (subscribers) consume only those messages that are of interest to them. This model helps create a highly efficient and flexible communication system where information is only transmitted when necessary.

In the context of Ontropy, the pub-sub model enables efficient communication within the state channels. Each state channel can be thought of as a topic in the pub-sub model, and the participants in the state channel are the subscribers. The participants publish their actions to the channel, and all other participants receive these messages. This allows all participants in a state channel to stay in sync with each other without having to interact with every other participant individually.

As for cryptography, Ontropy integrates advanced cryptographic primitives seamlessly with its networking layer. The integration of BLS and Schnorr signatures with the libp2p networking layer means that messages in the state channel are not just efficiently transmitted, but also securely verified. This integration makes Ontropy's solution robust against malicious actors and prevents unauthorized actions within the state channels.

The inclusion of a versatile cryptographic arsenal allows the system to generate proofs for all the actions within a state channel off-chain, ensuring that the state transitions are valid and that this could be done for a wide number of use cases. These proofs can then be submitted to the main chain when the state channel is ready to be closed or in case of a dispute. This approach reduces the number of transactions committed to the blockchain, resulting in lower gas fees and faster processing times.

By combining libp2p and advanced cryptographic primitives, Ontropy has not only proposed an effective scaling solution but also made it easily accessible and implementable. Whether it's the flexibility of state channels or the robustness of cryptographic signatures, Ontropy provides a toolkit that effectively addresses the challenges of building scalable, decentralized applications.

5.1 Schnorr Signatures

The Schnorr signature scheme, proposed by Claus-Peter Schnorr, forms a significant part of the Ontropy protocol's cryptographic toolbox. The use of Schnorr signatures contributes to the Protocol's mission of developing efficient and secure decentralized systems by offering several unique features that boost its cryptographic robustness and computational efficiency.

Schnorr signatures are renowned for their simplicity, security, and scalability, mainly due to the linear structure of the signing and verification equations. The mathematical properties of Schnorr signatures also allow for secure signature aggregation, a feature that can greatly reduce transaction sizes and thus improve the overall throughput of a blockchain. In contrast to ECDSA, Schnorr signatures make the Ontropy protocol capable of handling multi-signature transactions more efficiently. However, the implementation of Schnorr signatures must address the "rogue-key" attack vulnerability. This attack vector emerges in multi-signature scenarios, where one dishonest participant can manipulate their key to gain control over the joint key, and thus forge signatures on behalf of the whole group. To mitigate this risk, the Ontropy protocol includes careful key setup procedures to prevent the creation of rogue keys.

Ontropy's implementation of Schnorr signatures is further enhanced with Ethereum's ECRECOVER opcode for signature verification. The ECRECOVER function uses elliptic curve cryptography to recover the signer of a message, thus establishing a direct link between an address and a signed message without revealing the private key.

In a usual Ethereum context, ECRECOVER is primarily used to verify ECDSA signatures, but the Ontropy protocol adapts it to verify Schnorr signatures. Specifically, we are using a special method that maps the Schnorr signature to an equivalent ECDSA signature that the ECRECOVER opcode can verify. This approach enables Schnorr signature verification with ECRECOVER, offering a massive reduction in gas costs, to around 6k gas per verification, compared to native Schnorr signature verification.

This ability to reduce transaction costs and improve efficiency is crucial to the Ontropy protocol, as it enables higher transaction throughput and makes interactions with the Ontropy network more economically viable for users. However, using ECRECOVER for Schnorr signature verification requires careful handling. In particular, we must ensure that the signature mapping does not introduce any security vulnerabilities or affect the signature's original security properties.

To guarantee this level of security and performance, all Schnorr signature operations, like ECDSA, are implemented in Rust within the Ontropy protocol. We enforce a policy of conducting all operations in constant time to minimize timing attack vectors and maintain predictable performance. Furthermore, the binaries are subject to rigorous benchmarking to ensure their efficiency and security in real-world use cases.

In conclusion, the inclusion of Schnorr signatures in the Ontropy protocol highlights our dedication to leveraging advanced cryptographic techniques for improved performance and security. By mitigating the rogue key attack vector, adapting ECRECOVER for efficient verification, and committing to stringent performance and security standards, we ensure that Ontropy can offer a highly secure and efficient blockchain protocol.

5.1.1 MuSig2

Schnorr signatures are undoubtedly effective and offer economically viable on-chain verification, but they do come with certain inherent security challenges. For instance, the aforementioned "rogue-key" attack is a notable concern. To overcome such vulnerabilities and to further bolster the efficiency of the Ontropy protocol, we leverage a powerful multi-signature scheme known as MuSig2. MuSig2, a successor to the original MuSig scheme, is a novel protocol that offers secure, standardized, and efficient Schnorr multi-signatures. It was initially conceived for implementation within the Bitcoin network and has since gained widespread recognition for its security and efficiency enhancements.

By deploying Musig2 within the Ontropy protocol, we effectively address Schnorr's rogue-key attack vulnerability. In a multi-signature context, MuSig2 ensures that even if a participant attempts to alter their key during the setup phase, it would not affect the final aggregated key. Thus, it offers protection against this key manipulation and secures the signature process against forgery.

MuSig2's robustness stems from its adherence to strict security assumptions. It ensures non-interactivity by allowing participants to exchange only two rounds of messages, which significantly mitigates interactive attack vectors. This non-interactivity is based on the assumption of a secure communication channel between the signers, which makes Musig2 secure against concurrent sessions.

Furthermore, it is worth noting that the MuSig2 protocol is built on the Discrete Logarithm Assumption (DLA) and the Random Oracle Model (ROM). DLA is a cryptographic assumption that states that it is computationally hard to compute the discrete logarithm in a cyclic group, given the base and the result of the exponentiation. The ROM, on the other hand, assumes the existence of a hypothetical, perfectly random function, which is used for modeling hash functions in cryptographic proofs. Both these assumptions are well-established and have been extensively vetted in the field of cryptography.

Just like our Schnorr and ECDSA operations, the implementation of MuSig2 is carried out in Rust, ensuring optimal performance and top-tier security. Again, we enforce a constant time execution policy to avert timing attack vectors and regularly benchmark the resulting binaries to ensure top-notch performance and reliability in real-world deployment scenarios.

Musig2 forms an integral part of the Ontropy protocol's cryptographic architecture. By effectively tackling Schnorr's security vulnerabilities, enabling non-interactive multi-signature transactions, and offering optimal performance and security, MuSig2 further reinforces Ontropy's commitment to providing an advanced, secure, and efficient decentralized platform.

5.1.2 FROST

While Musig2 addresses several challenges in multi-signature schemes and offers numerous benefits, we recognize the value of diverse cryptographic solutions within the Ontropy protocol. This brings us to Flexible Round-Optimized Schnorr Threshold Signatures (FROST2), another high-performing multi-signature scheme that offers its own unique advantages. FROST2 becomes especially relevant when considering future requirements for threshold signatures, which are not supported by MuSig2. Threshold signatures allow a subset of a group to sign on behalf of the whole group, which can be crucial in certain transaction or governance scenarios. In the case of Ontropy, incorporating threshold signatures would provide flexibility for future development paths and use cases.

Furthermore, FROST2 facilitates the swapping of public keys or changes to the multi-signature scheme without necessitating an on-chain transaction. This feature enhances the dynamic nature of Ontropy, offering superior adaptability to our users.

While MuSig2 operates safely within non-concurrent settings, its security in concurrent settings may raise concerns, as the complexity of proving concurrent security can be challenging. FROST2 comes to the fore here, as it has robust security mechanisms that are more suited to concurrent environments.

On the other hand, it's important to acknowledge the situations where MuSig2 might be preferable over FROST2. For instance, if only n-of-n signatures are needed, Musig2 can optimize this operation. Also, for those wishing to avoid an extra communication round required by FROST2 for key generation and aggregation, MuSig2 could be the preferred option. Lastly, the simpler and more standard cryptographic model employed by Musig2 might be more appealing for certain implementations.

It's important to note that the choice between MuSig2 and FROST2 doesn't have to be exclusive. They can coexist within the same scheme, allowing for nested setups, such as a MuSig2 setup within a FROST2 setup, or vice versa. This hybrid approach not only provides robust security but also allows for a smooth transition between the two schemes if needed. For instance, MuSig2 keys can be converted to FROST2 keys without changing the aggregate public key, ensuring seamless operation and user experience.

Like our other cryptographic operations, FROST2 is implemented in Rust within the Ontropy protocol, optimizing security and performance. Our commitment to constant time execution and regular benchmarking of the resulting binaries also applies to FROST2, ensuring a secure, efficient, and performant implementation.

FROST2 provides yet another layer of advanced cryptographic architecture within the Ontropy protocol. With its unique strengths and flexibility to coexist with MuSig2, FROST2 reinforces Ontropy's commitment to robust, adaptable, and future-proof cryptographic solutions.

While both MuSig2 and FROST2 represent innovative strides in multi-signature cryptographic technologies, each of these protocols offer a unique set of advantages that could be leveraged to meet varying needs within the Ontropy ecosystem. A hybrid approach of integrating MuSig2 and FROST2 is currently under investigation, which could maximize the benefits of both. At a high level, this hybrid methodology of combining MuSig2 and FROST2 opens up multiple opportunities: 1. The aggregate public key remains consistent when converting MuSig2 keys to FROST2 keys. This provides a frictionless transition between the two protocols, a key factor in maintaining a smooth user experience.

2. Implementing a "broadcast channel" is simplified by utilizing MuSig2 to sign a hashed list of coefficient commitments. This results in a much less complex protocol that's easier to manage and maintain.

3. The hybrid approach leverages existing MuSig2 APIs. This aids in streamlin-

ing the code implementation process while ensuring regular upkeep.

Here is how the modified FROST protocol operates in this hybrid model:

The first step involves generating a MuSig2 key. Following this, participants create random polynomial coefficients, with the first coefficient being the product of their individual MuSig2 private key and the key aggregation coefficient. The result is a unified aggregate public key for both FROST2 and MuSig2, as the FROST2 aggregate public key is the sum of the commitments to each participant's first polynomial coefficient. The additional benefit is that the participants don't need to distribute a proof of knowledge of their first coefficient (as required in the original FROST protocol), as the MuSig2 protocol and the key aggregation coefficients effectively thwart the rogue key attack.

Participants then share their respective coefficient commitments and MuSig2 nonce commitment pairs. Once all the coefficient commitments are received, a hash of the list is created and signed using MuSig2. The partial MuSig2 signatures are aggregated, and a valid signature verifies that each participant has received the same commitments, thus effectively simulating a broadcast channel. This process completes the key generation phase.

At this stage, every participant possesses a FROST2 share usable for threshold signing. However, for nonce generation, the protocol employs MuSig2 nonces, which aligns with FROST2 nonce generation in the way they use pairs of nonces and nonce commitments.

The combination of MuSig2 and FROST2 is not merely a theoretical proposition. The practical implementation of this methodology can be found in the open-source secp256k1-zkp library.

The hybridization of MuSig2 and FROST2 within the Ontropy protocol serves as a demonstration of the commitment to adopting robust, flexible, and adaptable cryptographic solutions. This approach does not only streamline the operations but also ensures that Ontropy stays future-proof, ready to accommodate the needs of evolving cryptographic requirements and scenarios.

At Ontropy, we are committed to delivering secure, efficient, and innovative solutions backed by a carefully crafted development timeline. For the first phase of Ontropy v. 1.0, we have chosen to implement MuSig2 within our Virtual Rollup protocol. MuSig2 brings forth notable efficiency and simplicity, which makes it a perfect fit for the initial launch. Its interoperability with Bitcoin's ecosystem, as well as its easy on-chain verification, are features that we are keen on leveraging to provide our users with an optimal experience.

However, Ontropy's roadmap does not stop at MuSig2. In recognition of the distinct advantages that FROST2 provides, particularly its support for threshold signatures and the flexibility it brings to the table in terms of public key swap-outs and multi sig scheme changes, we intend to explore its integration into Ontropy in later versions.

We aim to implement a hybrid MuSig2 and FROST2 system, which will not only uphold our commitment to security and efficiency but also enable us to provide an even wider range of services and capabilities to our users. This plan for the future embodies our dedication to continuous innovation, development, and adaptation in response to the ever-evolving landscape of blockchain technology.

Our chosen path-beginning with MuSig2 for Ontropy v. 1.0 and later transitioning to a hybrid model-provides us with a roadmap that ensures both robust functionality for today and flexibility for the future.

5.1.3 Schnorr Scheme Comparison and Development Schedule

While both MuSig2 and FROST2 represent innovative strides in multi-signature cryptographic technologies, each of these protocols offer a unique set of advantages that could be leveraged to meet varying needs within the Ontropy ecosystem. A hybrid approach of integrating MuSig2 and FROST2 is currently under investigation, which could maximize the benefits of both. At a high level, this hybrid methodology of combining MuSig2 and FROST2 opens up multiple opportunities:

1. The aggregate public key remains consistent when converting MuSig2 keys to FROST2 keys. This provides a frictionless transition between the two protocols, a key factor in maintaining a smooth user experience.

2. Implementing a "broadcast channel" is simplified by utilizing MuSig2 to sign a hashed list of coefficient commitments. This results in a much less complex protocol that's easier to manage and maintain.

3. The hybrid approach leverages existing MuSig2 APIs. This aids in streamlining the code implementation process while ensuring regular upkeep.

Here is how the modified FROST protocol operates in this hybrid model:

The first step involves generating a MuSig2 key. Following this, participants create random polynomial coefficients, with the first coefficient being the product of their individual MuSig2 private key and the key aggregation coefficient. The result is a unified aggregate public key for both FROST2 and MuSig2, as the FROST2 aggregate public key is the sum of the commitments to each participant's first polynomial coefficient. The additional benefit is that the participants don't need to distribute a proof of knowledge of their first coefficient (as required in the original FROST protocol), as the MuSig2 protocol and the key aggregation coefficients effectively thwart the rogue key attack. Participants then share their respective coefficient commitments and MuSig2 nonce commitment pairs. Once all the coefficient commitments are received, a hash of the list is created and signed using MuSig2.

The partial MuSig2 signatures are aggregated, and a valid signature verifies that each participant has received the same commitments, thus effectively simulating a broadcast channel. This process completes the key generation phase. At this stage, every participant possesses a FROST2 share usable for threshold signing. However, for nonce generation, the protocol employs MuSig2 nonces, which aligns with FROST2 nonce generation in the way they use pairs of nonces and nonce commitments.

The combination of MuSig2 and FROST2 is not merely a theoretical proposition. The practical implementation of this methodology can be found in the open-source secp256k1-zkp library.

The hybridization of MuSig2 and FROST2 within the Ontropy protocol serves as a demonstration of the commitment to adopting robust, flexible, and adaptable cryptographic solutions. This approach does not only streamline the operations but also ensures that Ontropy stays future-proof, ready to accommodate the needs of evolving cryptographic requirements and scenarios.

At Ontropy, we are committed to delivering secure, efficient, and innovative solutions backed by a carefully crafted development timeline. For the first phase of Ontropy v. 1.0, we have chosen to implement MuSig2 within our Virtual Rollup protocol. MuSig2 brings forth notable efficiency and simplicity, which makes it a perfect fit for the initial launch. Its interoperability with Bitcoin's ecosystem, as well as its easy on-chain verification, are features that we are keen on leveraging to provide our users with an optimal experience.

However, Ontropy's roadmap does not stop at MuSig2. In recognition of the distinct advantages that FROST2 provides, particularly its support for threshold signatures and the flexibility it brings to the table in terms of public key swap-outs and multi sig scheme changes, we intend to explore its integration into Ontropy in later versions.

We aim to implement a hybrid MuSig2 and FROST2 system, which will not only uphold our commitment to security and efficiency but also enable us to provide an even wider range of services and capabilities to our users. This plan for the future embodies our dedication to continuous innovation, development, and adaptation in response to the ever-evolving landscape of blockchain technology.

Our chosen path-beginning with MuSig2 for Ontropy v. 1.0 and later transitioning to a hybrid model provides us with a roadmap that ensures both robust functionality for today and flexibility for the future.

5.2 BLS Signatures

With the foundations firmly established via the sophisticated use of MuSig2 and FROST2, the Ontropy protocol is ready to ascend to the next level of advanced cryptographic operations. We turn our focus to the integration of the Barreto-Lynn-Scott (BLS) signature scheme, specifically BLS12-381, augmenting the dynamic and adaptable nature of Ontropy. Adopting BLS12-381 is not a mere additive measure to our cryptographic repertoire but a transformative step that supercharges Ontropy's ability to meet future cryptographic demands and challenges.

Central to BLS12-381's promise is its facility for independent signatures. Unlike MuSig2 and FROST2, the BLS signature scheme allows for individual signatures to be independently validated, offering increased security and flexibility in multi-signature operations. This feature significantly enhances Ontropy's robustness, catering to more complex transactional or governance scenarios. BLS12-381 also shines in its support for threshold signatures. This signature scheme permits a subset of a group to authentically sign on behalf of the whole, an attribute essential for accommodating varying use cases in a dynamic blockchain environment.

Moreover, BLS12-381 brings to the table an impressive pairing-friendly property, laying the groundwork for more intricate Zero-Knowledge Proofs (ZKP). Currently, Ontropy leverages efficient and lightweight ZKP mechanisms, optimizing the privacy-security balance without straining computational resources. As we envision the evolution of Ontropy, however, the need for more sophisticated privacy measures may arise, making BLS12-381's support for such advanced ZKPS a vital asset.

Despite BLS12-381's unique advantages, we recognize that there may be contexts where MuSig2 or FROST2 would be a more suitable fit. Whether it's the need for n-of-n signatures, an inclination to evade the extra communication round for key generation and aggregation, or a preference for a simpler cryptographic model, the protocol allows users to choose.

Notably, the integration of BLS12-381 does not necessitate an exclusive cryptographic commitment. Rather, BLS12-381 can coexist with MuSig2 and FROST2 within the same protocol, facilitating nested setups and thus maximizing the flexibility and adaptability of Ontropy.

The addition of BLS12-381 into Ontropy's cryptographic armory signals a marked progression in the protocol's architectural evolution. BLS12-381, with its unique strengths and compatibility with MuSig2 and FROST2, reaffirms Ontropy's unwavering dedication to advanced, adaptable, and resilient cryptographic solutions.

The respective merits of MuSig2, FROST2, and BLS12-381 each contribute to Ontropy's advanced cryptographic system. The inclusion of BLS12-381, in particular, offers many novel opportunities, solidifying Ontropy's commitment to a secure, efficient, and forward-thinking cryptographic landscape. The strategic integration of BLS12-381 ensures the system's readiness for the cryptographic demands of the future, facilitating a smooth and seamless transition among MuSig2, FROST2, and BLS12-381. The result is an optimized user experience, unburdened by complex technical transitions. Ontropy stands as a testament to the powerful combination of robust, flexible, and adaptable cryptographic systems. Our strategic incorporation of BLS12-381 not only streamlines operations but also equips Ontropy with the capabilities to meet future cryptographic necessities.

BLS12-381's integration into the Ontropy protocol adheres to our commitment to providing secure, high-performing, and efficient cryptographic solutions. Our pursuit of constant-time execution and routine benchmarking extends to BLS12-381, ensuring that our users benefit from a secure and performant implementation.

Ontropy's development roadmap acknowledges the unique advantages of each cryptographic protocol-MuSig2, FROST2, and BLS12-381. Our initial choice of MuSig2, with its simplicity and efficiency, serves as an ideal starting point for the first phase of Ontropy. However, our vision extends beyond this foundational stage.

Recognizing the need for a more dynamic and adaptable cryptographic architecture, we plan to progressively incorporate FROST2 and BLS12-381 into future iterations of Ontropy. This roadmap ensures that Ontropy remains equipped for today's needs while staying ready for the challenges of tomorrow. As Ontropy continues to evolve and adapt, we remain committed to delivering secure, innovative, and user-oriented solutions, offering not just robust functionality but also future-proof flexibility. With the strategic integration of BLS12-381 into our cryptographic framework, Ontropy continues to be at the forefront of cutting-edge blockchain technology, ready to meet the ever-evolving demands of the cryptographic landscape.

5.3 Pedersen Hash

While SHA-3 provides robust general-purpose hashing, Pedersen hash has been chosen for its specific properties that align with the requirements of our protocol. Pedersen hash, an essential component in many privacy-preserving cryptographic protocols, is primarily used for creating commitments. Unlike SHA-3, a Pedersen hash operates within the confines of an elliptic curve, thereby enabling commitments that are both hiding (it is computationally hard to reveal the committed value without knowing the blinding factor) and binding (once a value has been committed, it cannot be changed). In the context of Ontropy, Pedersen hash is employed in the generation of Pedersen commitments, enabling a prover to commit to a chosen value while keeping it hidden and preventing the alteration of the value post commitment. This is particularly useful in zero-knowledge proof contexts such as during random number generation and the obfuscation of transaction details. By leveraging the properties of Pedersen hashes in conjunction with elliptic curves, Ontropy is capable of realizing complex cryptographic operations in an efficient and secure manner. The ability to construct commitments using Pedersen hashes on elliptic curves significantly enhances the protocol's privacy and security aspects.

5.4 Foque-Tibouchi

In the Ontropy protocol, the hashing functions of SHA-3 and Pedersen hold critical importance, yet there is another component that plays a pivotal role in maintaining the security and functionality of the system: the Fouque-Tibouchi (FT) method. The method is named after Jean-Sébastien Coron, Pierrick Méaux, David Naccache, and Mehdi Tibouchi who proposed it in their work. This mechanism is particularly essential when it comes to the domain of hash-to-curve algorithms. The nuances of the FT method and its divergence from SHA-3 and Pedersen hashes significantly elevate the Ontropy protocol's capabilities.

The FT method is a hash-to-curve a deterministic algorithm that allows arbitrary input strings to be mapped to points on an elliptic curve. It is used in many cryptographic processes such as the construction of digital signatures, key exchange protocols, and encryption algorithms.

Unlike the SHA-3 and Pedersen hashes, which are primarily utilized for securing data integrity and constructing commitments, respectively, the FT method focuses on creating valid and uniformly random points on an elliptic curve from any given input. It is deterministic, ensuring the same output for the same input, and it ensures that outputs are evenly distributed across the elliptic curve. This property is crucial in maintaining the security properties of many protocols that rely on the uniform randomness of points on elliptic curves.

An important feature of the FT method that sets it apart is its handling of exceptional cases. Unlike some other hash-to-curve algorithms, the FT method does not use rejection sampling (a common practice of rejecting certain outputs to achieve uniform distribution). Instead, it handles all input strings uniformly and deterministically, making it more efficient and easier to analyze for security properties. In the Ontropy protocol, the FT method enables a secure and efficient mapping of hash outputs to points on elliptic curves, a process that is critical for the protocol's advanced cryptographic operations, particularly those involving zero-knowledge proofs and secure multi-party computations.

In conclusion, the incorporation of the FT method alongside SHA-3 and Pedersen hash in the Ontropy protocol ensures a comprehensive and robust cryptographic scheme. While SHA-3 provides broad cryptographic security and interoperability, and Pedersen hash offers privacy-preserving commitments, the FT method enhances the protocol with secure and efficient hash-to-curve functionality. The harmonious integration of these different cryptographic techniques is what equips the Ontropy protocol with its advanced capabilities and robustness.

FT is currently a candidate method for Ontropy Virtual Rollup Protocol.

5.4.1 Elligator2

As we go deeper into the inner workings of the Ontropy protocol, it becomes necessary to introduce another pivotal cryptographic algorithm: Elligator2. This particular mechanism, similar to Fouque-Tibouchi (FT) in functionality but distinct in its nuances, is key in the robust cryptographic architecture of Ontropy. Elligator2 is an ingenious method for the deterministic mapping of strings to points on an elliptic curve, much like FT. Developed by Bernstein, Hamburg, Krasnova, and Lange, Elligator2 plays an indispensable role in systems necessitating privacy since it facilitates constructions that are indistinguishable from random strings in the codomain. What sets Elligator2 apart is its unique ability to map any given string to a point on the curve in a way that is not only deterministic but also bijective. This bijection ensures that for each point on the curve, there's exactly one corresponding input string, and vice versa, a property not seen in FT. This injective and surjective mapping permits back-and-forth conversion between curve points and strings without loss of data, a trait that is desirable in certain cryptographic operations. Another distinguishing factor of Elligator2 is that it is designed to work with elliptic curves represented in Montgomery or Edwards form, which are often used in efficient implementations of elliptic curve cryptography due to their faster arithmetic operations. While both FT and Elligator2 aim to accomplish the same goal - to transform arbitrary strings into curve points - the use case and application largely determine the choice between the two. FT's approach is highly beneficial when uniformity and simplicity of analysis are paramount, whereas Elligator2 shines when bijective mapping and working with Edwards or Montgomery curves are critical requirements. In the context of the Ontropy protocol, Elligator2 further bolsters the cryptographic operations involving hash-to-curve transformations. It works hand in hand with the FT method, SHA-3, and Pedersen hash to provide a comprehensive cryptographic backbone that efficiently meets the protocol's privacy, security, and interoperability objectives. By employing Elligator2 alongside these other techniques, Ontropy capitalizes on the unique strengths of each, thereby optimizing its cryptographic scheme. Elligator2 is currently a candidate method for Ontropy Virtual Rollup Protocol.

5.4.2 Kate Commitments

KZK commitments are an important moving part in an Ontropy Virtual Rollup solutions. It's main benefits include efficiency in batch verifications and creating short proof. Unlike Pedersen commitment scheme, you can prove the correctness of a polynomial's evaluations at many points with a proof size that does not increase with the number of points. Let's dive deeper in how do they work and how we use them. Here's how we represent a Kate commitment for a polynomial f: $C_f = g^{f(s)}$

In this equation, g is a generator of a cyclic group of prime order, s is a selected secret, and the caret symbol (ˆ) represents exponentiation within the group. To expose the value of polynomial f at a particular position x, we provide $f(x)$ and a proof $\pi = g^{f(s)-xs}$. Then, the verifier checks the following equation: [...]

$g^{f(x)} = C_f/\pi^x$

Now, let's pivot to Merkle trees. Merkle trees are an efficient way to verify the integrity of large data sets. For a list of data items d[1], ..., d[n], we start by hashing each item to produce a list of leaf nodes: l[i] = H(d[i]) Next, for every pair of leaf nodes $l[2*i], l[2*i+1]$, a parent node is created:

$p[i] = H(l[2*i]||l[2*i+1])$

This procedure is repeated until we eventually obtain a single root node r. When contrasting these two cryptographic schemes, it's apparent that they are utilized for different purposes. Merkle trees validate the inclusion of specific data within a set without exposing the entire set. Conversely, Kate commitments allow us to commit to a polynomial and later reveal its values at certain points. Moreover, the size of proofs in Kate commitments remains constant, unlike the logarithmic size increase in Merkle trees, thereby making Kate commitments more efficient for certain applications. However, this efficiency comes with increased complexity in setup assumptions.

5.4.3 Kate Vector Commitments

The Kate commitment scheme, initially conceived as a polynomial commitment, actually presents an interesting utility as a vector commitment as well. Let's take a moment to recall that a vector commitment is a commitment to a vector $a_0, \ldots, a_{n-1}$, allowing one to validate that commitment to a_i for a given i. Remarkably, this can be emulated using the Kate commitment scheme. We start by defining $p(X)$ as a polynomial that satisfies $p(i) = a_i$ for all i. It's a known fact that such a polynomial exists, and it can be conveniently computed using the method of Lagrange interpolation, as shown:

$p(X) = \sum_{i=0}^{n-1} a_i \prod_{\substack{j=0 \\ j\neq i}}^{n-1} \frac{X-j}{i-j}$

With this polynomial in our arsenal, we can prove an arbitrary number of elements in the vector using merely a single group element! This feature is a significant advantage in terms of proof size, and it showcases the efficiency of the Kate commitment scheme.

When juxtaposed with Merkle trees, the efficiency becomes more apparent. A Merkle proof, after all, would necessitate $\log n$ hashes even to substantiate a single element. Therefore, the Kate commitment scheme offers a more compact and efficient solution for such tasks.

5.5 Perspective Timelocked Encryption Solution

Another promising approach that we are developing is based on a novel time-locked encryption scheme proposed in [8]. Time-lock encryption serves as a method for encrypting a message so that its decryption is only possible after a specified deadline. In the study, the authors introduce an innovative time-lock encryption scheme. Its primary advantage over earlier models is that receivers, even those with relatively low computational resources, can instantly decrypt the message once the deadline has passed. This is achievable without interacting with the sender, other receivers, or any trusted third party.

The proposed time-lock encryption solution is built upon the idea of computational reference clocks and an extractable witness encryption scheme. It is proposed to construct the computation reference clocks from a public blockchain. In the article, only the Proof Of Work chains are studied. The most significant result of the paper is that it demonstrates how to achieve a constant level of multilinearity for witness encryption through the use of SNARKs.

The scheme is based on Subset-Sum problem and achieves extractable security without depending on obfuscation. This particular model employs multilinear maps of any given order and operates independently of the specific implementations of multilinear maps. Ontropy Virtual Rollup solution employs the timelock encryption scheme over the public blockchain to timelock the ephemeral keys. The smart contract events containing specific game scenarios are also used as a witness for decrypting the keys of the dropped out player. Let's describe how this time lock encryption scheme could work in more detail.

5.5.1 Computational reference clocks

In a timelock encryption scheme the public blockchain is viewed as a computational reference clock. The PoW blockchain performs an iterative, very large-scale, public computation, where validators are contributing significant computational to the gradual extension of the block chain. Essentially, this blockchain consists of a sequence of hash values $B_1, \ldots, B_\tau$ that satisfy certain conditions. [1] $B_{\tau+1}$ is appended to the chain. Thus, the blockchain can serve as a reference clock, where the current length τ of the chain tells the current "time", and there are about 12 seconds between each "clock tick" in the Ethereum network. Ontropy is currently exploring different ways of generalizing the PoW solution proposed to a PoS blockchains.

5.5.2 Witness encryption

To construct the time-lock encryption from computational reference clocks witness encryption is employed. Witness encryption for all NP-relations was introduced by Garg et al. [9]. Existing witness encryption schemes are based on multilinear maps or obfuscation. A witness encryption scheme is associated with an NP-relation R. For $(x, w) \in R$ we say that x is a "statement" and w is a "witness". A witness encryption scheme for relation R allows to encrypt a message m with respect to statement x as $c \overset{\$}{\leftarrow}$ WE.Enc (x, m). Any witness w which satisfies $(x, w) \in R$ can be used to decrypt this ciphertext c as $m = \mathrm{WE} \cdot \mathrm{Dec}(c, w)$. A statement x could be viewed as a "public key", such that any witness w with $(x, w) \in R$ can be used as a corresponding "secret key".

A secure witness encryption scheme essentially guarantees that no adversary is able to learn any non-trivial information about a message encrypted for statement x, unless it already "knows" a witness w for $(x, w) \in R$. Witness encryption schemes with this property are called extractable [12, 13, 14]. The idea of extractable security was first proposed in [14], along with a candidate construction, but there are no known constructions with mathematical proof of extractable security.

5.5.3 From witness encryption to time-lock encryption

The key idea behind the scheme proposed in [8] is to combine a computational reference clock with witness encryption. For that NP-relation R is defined such that

(1) For $x \in \mathbb{N}$, statements have the form 1^x, that is, x in unary representation.

(2) Any valid block chain $w = (B_1, \ldots, B_x)$ of length at least x is a witness for statement 1^x, that is $(1^x, w) \in R$.

Let (WE.Enc, WE.Dec) be a witness encryption scheme for this particular relation R. Suppose the current state of the Bitcoin blockchain is $B_1, \ldots, B_\tau$. Then the block chain contains a witness w for $(1^x, w) \in R$ for all $x \leq \tau$. The Bitcoin blockchain is public. Therefore everybody is immediately able to decrypt any ciphertext $c \overset{\$}{\leftarrow} \mathrm{WE.Enc}\ (1^x, m)$ with $x \leq \tau$, by using the witness from the public block chain as the "decryption key".

The reference clock does not have to start with the genesis block of the blockchain. Clock's initial state w_0 could be set to be the latest block. In addition, the witness does not have to include all transaction data of the blockchain, as the hash chain of the block headers is sufficient for decryption.

Security of this construction Let $c = \mathrm{WE.\,Enc}\,(1^x, m)$ be a ciphertext with $x > \tau$. Under the assumption that the witness encryption scheme is secure, we will show that an adversary has only two possibilities to learn any non-trivial information about m. (1) The adversary waits until the public blockchain has reached length x. Then the chain contains a witness w for $(1^x, w) \in R$, which immediately allows to decrypt. However, note that then not only the adversary, but also everybody else is able to decrypt, by reading w from the public blockchain and computing $m = \mathrm{WE.\,Dec}(c, w)$. Speaking figuratively, "the time-lock has opened".

(2) The adversary tries to "put forward" the computational reference clock provided by the blockchain, by computing the missing blocks $B_{\tau+1}, \ldots, B_x$ of the chain secretly on its own, faster than the public computation performed by the collection of all miners in case of PoW. Note that this means that the adversary would have to outperform the huge computational resources gathered in the PoW chain.

5.5.4 Using SNARKs to reduce multilinearity level for witness encryption

The size of the ciphertext and the running time for encryption and decryption is linear in terms of the size of witness. Usually the linear complexity is very natural and efficient. Unfortunately, this is not good enough for witness encryption because all the instantiations of witness encryption are based on multilinear maps [15]. The research on multilinear maps is still in its infancy and is not yet practical regardless of recent lines of attacks. Most importantly, the existing implementations for multilinear maps are not compact, that is, the size of the group elements is polynomial in the multilinearity level (i.e., the maximum number of pairings). The multilinearity level is polynomial in the length of the witness. To mitigate this issue, a novel scheme is proposed. It's idea is to implement time-lock encryption by using SNARKs together with witness encryption.

Instead of directly encrypting with an instance x in witness encryption, the idea is to encrypt with SNARKs verification procedure for the statement $(x, w) \in R$. It is shown that using SNARKs can achieve constant multilinearity level regardless of the instance and witness, rather than the previous linear complexity.

5.5.5 Extractable witness encryption

A new extractable witness encryption scheme based on a special SUBSET- SUM (described below). To encrypt with any NP language, we present a reduction from the NP-complete CNF-SAT to our special SUBSETSum problem. We prove the extractable security of this construction in the Idealised Graded Encoding Model. To the best of our knowledge, this is the first construction of witness encryption to achieve extractable security, without the use of obfuscation.

We use a variant of multilinear maps [15] where the groups are indexed by the integer vectors, which allows us to efficiently encode an instance of the special SUBSETSUM problem. Suppose a u-linear map on groups $\{\mathbb{G}_{\mathbf{W}}\}_{\mathbf{w}}$ with $\mathbf{w} \leq \mathbf{u}$ (component-wise comparison). The pairing operation $\mathbf{e}_{\mathbf{w},\mathbf{w}'}$ maps $\mathbb{G}_{\mathbf{w}} \times \mathbb{G}_{\mathbf{w}'}$ into $\mathbb{G}_{\mathbf{w}+\mathbf{w}'}$ with $\mathbf{w} + \mathbf{w}' \leq \mathbf{u}$ by computing $\mathbf{e}_{\mathbf{w},\mathbf{w}'}\left(g^a_{\mathbf{w}}, g^b_{\mathbf{w}'}\right) = g^{ab}_{\mathbf{w}+\mathbf{w}'}$

The special SUBSET- SUM problem is: given a multi-set of positive integer vectors $\Delta = \{(\mathbf{v}_i : \ell_i)\}_{i \in I}$ where $(\mathbf{v}_i : \ell_i)$ means $\mathbf{v}_i$ occurs ℓ_i times in the multiset and a target sum-vector $\mathbf{s}$ such that $(\ell_i + 1)\,\mathbf{v}_i \not\leq \mathbf{s}$ and $\mathbf{v}_i$ are pairwise-distinct, to decide whether there exists a subset of Δ that can sum up to $\mathbf{s}$. The side condition $(\ell_i + 1)\,\mathbf{v}_i \not\leq \mathbf{s}$ is to guarantee the encoding of each integer vector $\mathbf{v}_i$ can only be used for at most ℓ_i times, in order to keep consistency between the encoding of the SUBSET-SUM and the original SUBSET-SUM problem. In multilinear maps, the vectors in $\Delta = \{(\mathbf{v}_i : \ell_i)\}_{i \in I}$ are encoded as $\left\{g^{\alpha^{\mathbf{v}_i}}_{\mathbf{v}_i}\right\}_{i \in I}$ and the target vector is encoded as $g^{\alpha^{\mathbf{s}}}_{\mathbf{s}}$. Suppose the subset-sum exists, that is $\sum_{i \in I} b_i \mathbf{v}_i = \mathbf{s}$ with b_i positive integers and $b_i \leq \ell_i$. Then we compute the encoding of the target sum as below

$$g^{\alpha^{\mathbf{s}}}_{\mathbf{s}} = \mathsf{e}(\underbrace{g^{\alpha^{\mathbf{v}_1}}_{\mathbf{v}_1}, \cdots, g^{\alpha^{\mathbf{v}_1}}_{\mathbf{v}_1}}_{b_1}, \underbrace{g^{\alpha^{\mathbf{v}_2}}_{\mathbf{v}_2}, \cdots, g^{\alpha^{\mathbf{v}_2}}_{\mathbf{v}_2}}_{b_2}, \cdots, \underbrace{g^{\alpha^{\mathbf{v}_{|I|}}}_{\mathbf{v}_{|I|}}, \cdots, g^{\alpha^{\mathbf{v}_{|I|}}}_{\mathbf{v}_{|I|}}}_{b_{|I|}})$$

In this way, each vector $\mathbf{v}_i$ only needs to be encoded once and the multiplication of the same encoding is in logarithm time which gains efficiency. To encrypt with any NP language, we present a reduction from CNF-SAT to our special SUBSET-SUM problem.

It is proven that encoding achieves extractability in the generic model of multilinear maps. The main technical detail is to construct an efficient extractor to extract a witness from the adversary's group operations. Constructing a witness extractor for the existing witness encryption schemes appears to be super-polynomial because of the expansion of adversary's query-polynomial. Soundness security is a special case of the extractable security.

Extractable witness encryption with arbitrary auxiliary inputs might be unattainable. A simple counter example is the following: suppose the sampler has $(x, w) \in R$, then the sampler obfuscates the decryption algorithm of witness encryption $z = \mathcal{O}(\text{WE}\,\text{Dec}(w, \cdot))$ and gives z to the adversary; the adversary can decrypt WE.Enc (x, m) by using the backdoor z and does not have to know w.

To circumvent this issue, authors define extractable security in an oracle model. The oracle is used to model the blockchain. The extractability is possible to achieve for most of the non-artificial oracles. In particular, when the oracle is instantiated with a decentralised cryptocurrency, this kind of backdoor is unlikely to exist since it is believed that no one can have a witness w in advance for each instance x.

5.5.6 Efficiency comparison

Assume a CNF formula has n variables and k clauses, and m literals. The ciphertext size of our witness encryption scheme is $2n + 2k + 1$ group elements. The evaluation time is $n + \mathcal{O}\left(k \log \frac{m}{2k}\right)$. The multilinearity level is $n + m - k$ and can be optimised to $n + \mathcal{O}\left(k \log \frac{m}{2k}\right)$.

The efficiency of encoding CNF-SAT depends on the reduction. The best reductions from CNF-SAT to EXACT- COVER is CNFSAT → 3-CNF-SAT → EXACT-COVER. However, the reduction from CNF-SAT to 3-CNF-SAT increases the number of variables and clauses by the size of the original CNF formula, that is $n' = \mathcal{O}(m)$ and $k' = \mathcal{O}(m)$ while m is $n \cdot k$ in the worst case. The reduction from 3-CNF-SAT to EXACTCOVER generates an instance of size $2n' + 7k' + 1$. Hence the encoding produces $\mathcal{O}(m)$ group elements asnd the evaluation time and multilinearity level are also $\mathcal{O}(m)$ with $m = n \cdot k$ in the worst case. There are three instantiations of witness encryption for CNF formulas. Two of them are specific to the composite order multilinear groups. The prime-order groups are usually more natural and result in simpler security assumptions. The conversion from composite-order construction to prime-order multilinear groups (or more generally, groups of arbitrary order) is very expensive and results in a ciphertext of $O\left(n^5 k^2\right)$ group elements.s

References

[1] Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman, Mental Poker. Technical Report MIT-LCS-TM-125, Massachusetts Institute of Technology

[2] Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman, Mental Poker, pages 37–43, 1981. The Mathematical Gardner

[3] Castellá-Roca, J., Sebé, F., Domingo-Ferrer, J. (2005). Dropout-Tolerant TTP-Free Mental Poker. In: Katsikas, S., López, J., Pernul, G. (eds) Trust, Privacy, and Security in Digital Business. TrustBus 2005. Lecture Notes in Computer Science, vol 3592. Springer, Berlin, Heidelberg.

[4] R. Lipton. How to cheat at mental poker. In Proc. AMS Short Course on Cryptography, 1981.

[5] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In In Proc. 14th ACM Symposium on Theory of Computing (STOC), pages 365–377, San Francisco, 1982. ACM

[6] O. Goldreich, S. Micali, A. Wigderson. How to play ANY mental game, STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing, 1987

[7] Kaleidoscope An Efficient Poker Protocol with Payment Distribution and Penalty Enforcement, Bernardo David, Rafael, Dowsley Mario Larangeira https://eprint.iacr.org/2017/899.pdf, 2018

[8] Jia Liu, Tibor Jager, Saqib A. Kakvi. How to build time-lock encryption , Bogdan Warinschi, 2017, https://eprint.iacr.org/2015/482.pdf 24, 26

[9] Garg S., Gentry C., Sahai A., Waters B.: Witness encryption and its applications. STOC '13, pp. 467–476 (2013) 25

[10] Zhandry M.: How to avoid obfuscation using witness PRFs. 2016. In: Proceedings of TCC

[11] Claude Cré*peau*, A Secure Poker Protocol that Minimizes the Effect of Player Coalitions, 1998, Conference: Advances in Cryptology, McGill University

[12] Bellare M., Hoang V.T.: Adaptive witness encryption and asymmetric password-based cryptography. Cryptology ePrint Archive, Report 2013/704, http://eprint.iacr.org/ (2013). 25

[13] Boyle E., Chung K.-M., Pass R.: On extractability obfuscation. In: Lindell, Yehuda (ed)., TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Berlin (2014). 25

[14] Goldwasser S., Kalai Y.T., Popa R.A., Vaikuntanathan V., Zeldovich N.: How to run turing machines on encrypted data. In: CRYPTO, pp. 536–553 (2013). 25, 26

[15] Boneh D., Silverberg A.: Applications of multilinear forms to cryptography. Contemp. Math. 324, 71–90 (2003). 26, 27