geometry

Enabling Decentralised Card Games with Zero Knowledge Proofs

Nicolas Mohnblatt
Nov. 22
nico@geometry.xyz

Once there were two “mental chess” experts who had become tired of their pastime.

> *“Let’s play ‘Mental Poker’, for variety” suggested one.*
>
> *“Sure” said the other. “Just let me deal!”*

from “Mental Poker”, Shamir, Rivest, Adleman, 1979

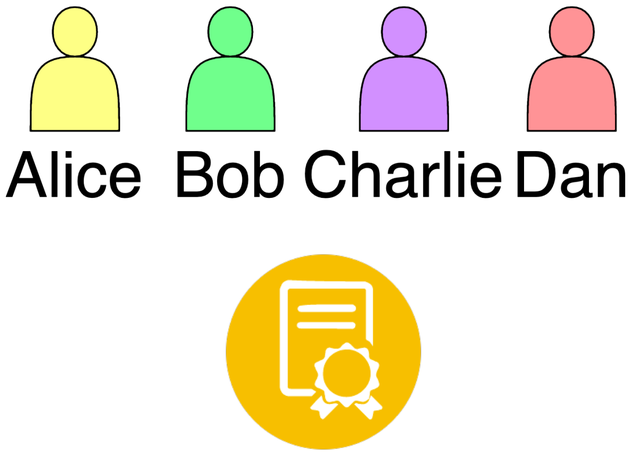
Agenda

1. Poker on blockchains: participants, requirements and challenges
2. Poker as a cryptographic problem
3. A round of Texas Hold’Em
4. 👹 Math (optional)
5. Implementation

Poker on Blockchains

Participants

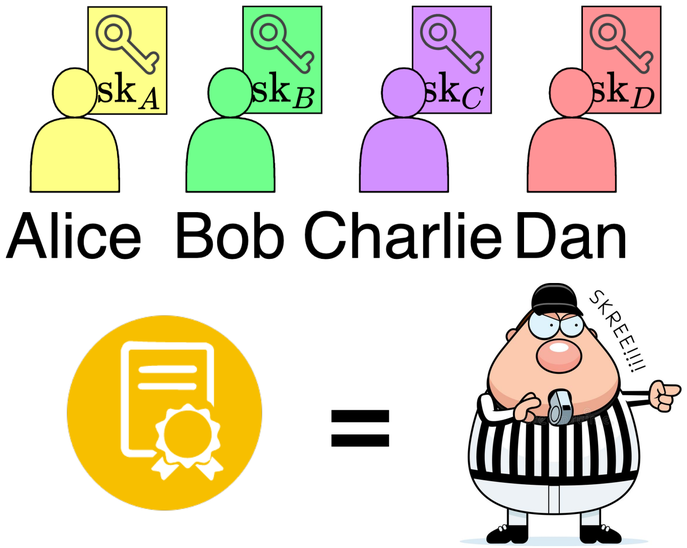
- players: untrusted, potentially malicious
- smart contract: transparent, will do as told but cannot keep a secret



Requirements

- hide cards
- reveal cards to all or individual players
- fair shuffle of the cards

Poker on Blockchains



Participants

- players: untrusted, potentially malicious
- smart contract: transparent, will do as told but cannot keep a secret

Requirements

- hide cards → Encryption
- reveal cards to all or individual players → (partial) decryption
- fair shuffle of the cards → All players participate

Mental Poker: a Cryptographic Challenge

Mental Poker

by Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman
MIT
Cambridge, Massachusetts 02139
November 29, 1978

Abstract

Can two potentially dishonest players play a fair game of poker without using any cards (e.g. over the phone)?

This paper provides the following answers:

(1) No. (Rigorous mathematical proof supplied.)

(1) Yes. (Correct & complete protocol given.)

Mental Poker: a Cryptographic Challenge

Mental Poker

by Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman
MIT
Cambridge, Massachusetts 02139
November 29, 1978

Abstract

Can two potentially dishonest players play a fair game of poker without using any cards (e.g. over the phone)?

This paper provides the following answers:

(1) No. (Rigorous mathematical proof supplied.)

(1) Yes. (Correct & complete protocol given.)

Mental Poker: a Cryptographic Challenge

Mental Poker

by Adi Shamir, Ronald L. Rivest, and Leonard M. Adleman
MIT
Cambridge, Massachusetts 02139
November 29, 1978

Abstract

Can two potentially dishonest players play a fair game of poker without using any cards (e.g. over the phone)?

This paper provides the following answers:

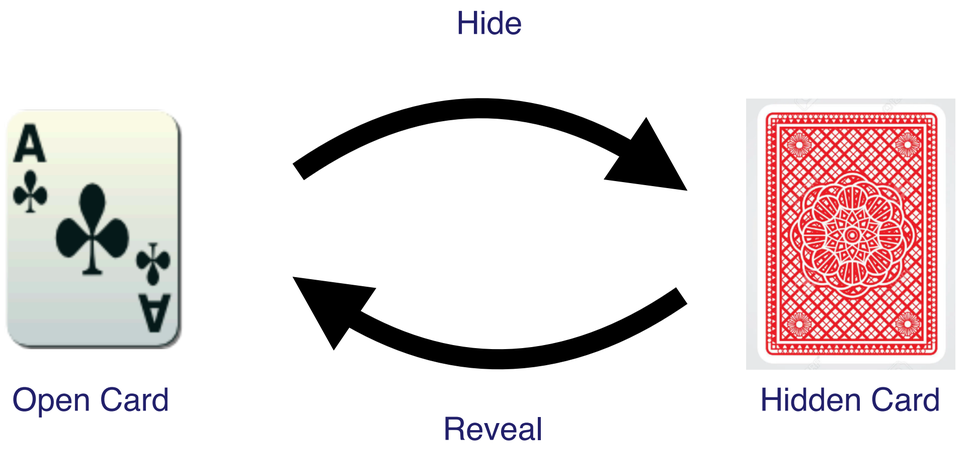
The blatant contradiction between our two results is real in that it is not due to any tricks or faults in either result. We will, in fact, leave to the reader the enjoyable task of puzzling out the differences in underlying assumptions that account for our contradictory results.

Mental Poker: a Cryptographic Challenge

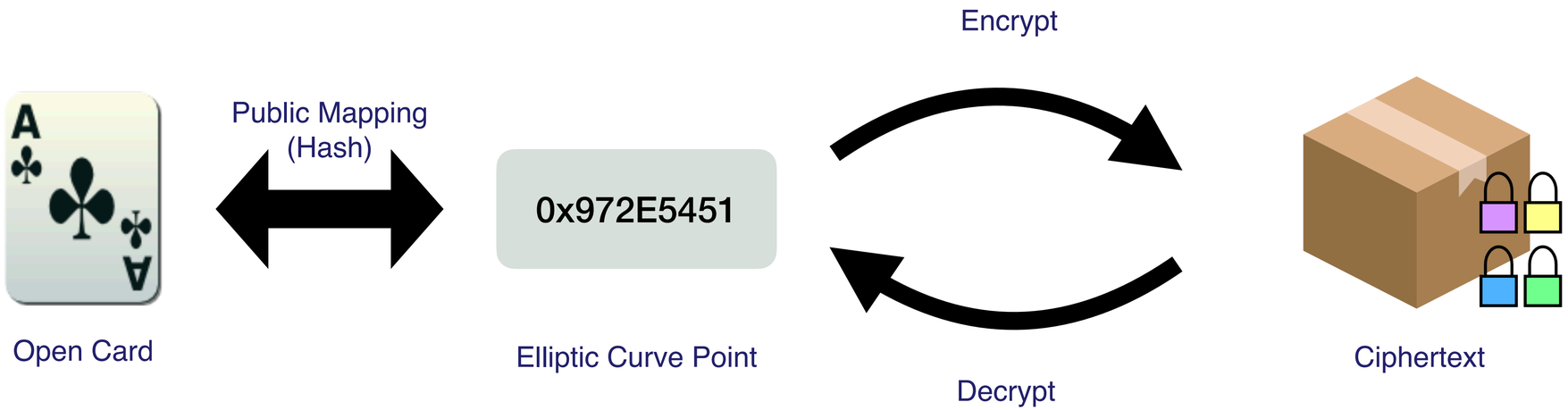
- How is this possible? Cryptography!
  You “can” learn cards but you’ll need to break encryption
- Modular solution by Barnett and Smart (from *Mental Poker Revisited,* 2003)

Verifiable threshold encryption + ZK proof of shuffle

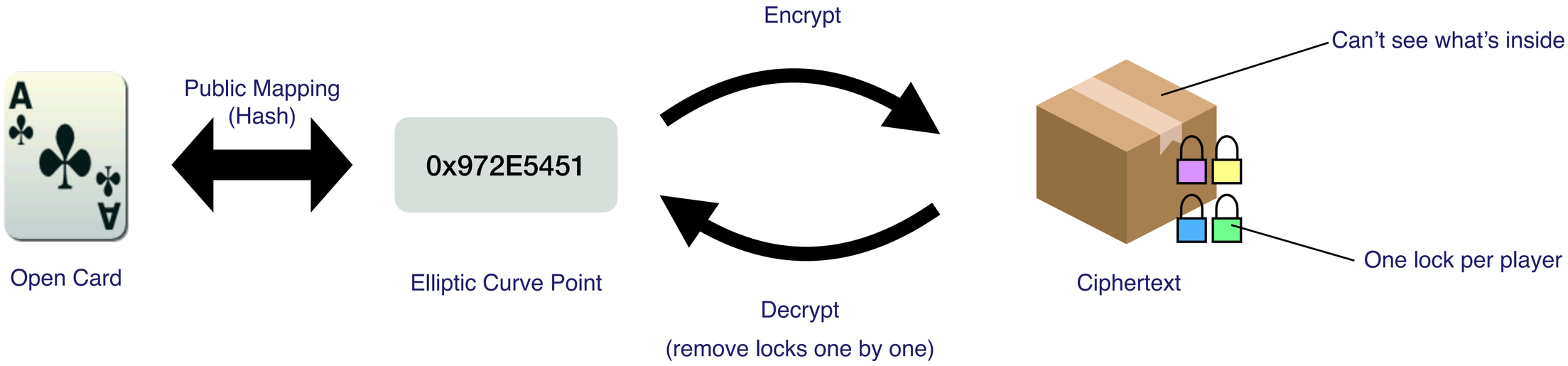
Card Lifecycle: Physical Card



Card Lifecycle: Digital Card



Card Lifecycle: Digital Card



Card Lifecycle: Digital Card

Open Card

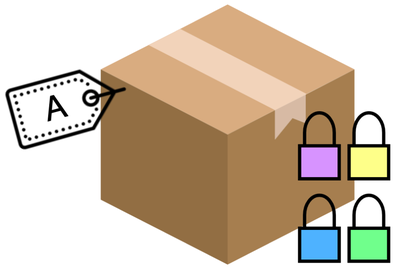
Public Mapping
(Hash)

0x972E5451

Elliptic Curve Point

Encrypt

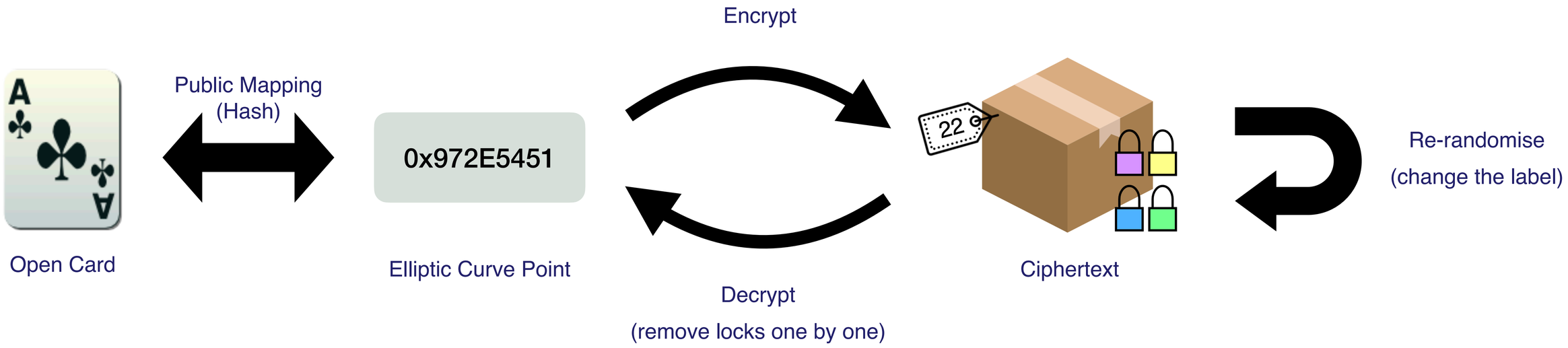
Decrypt
(remove locks one by one)



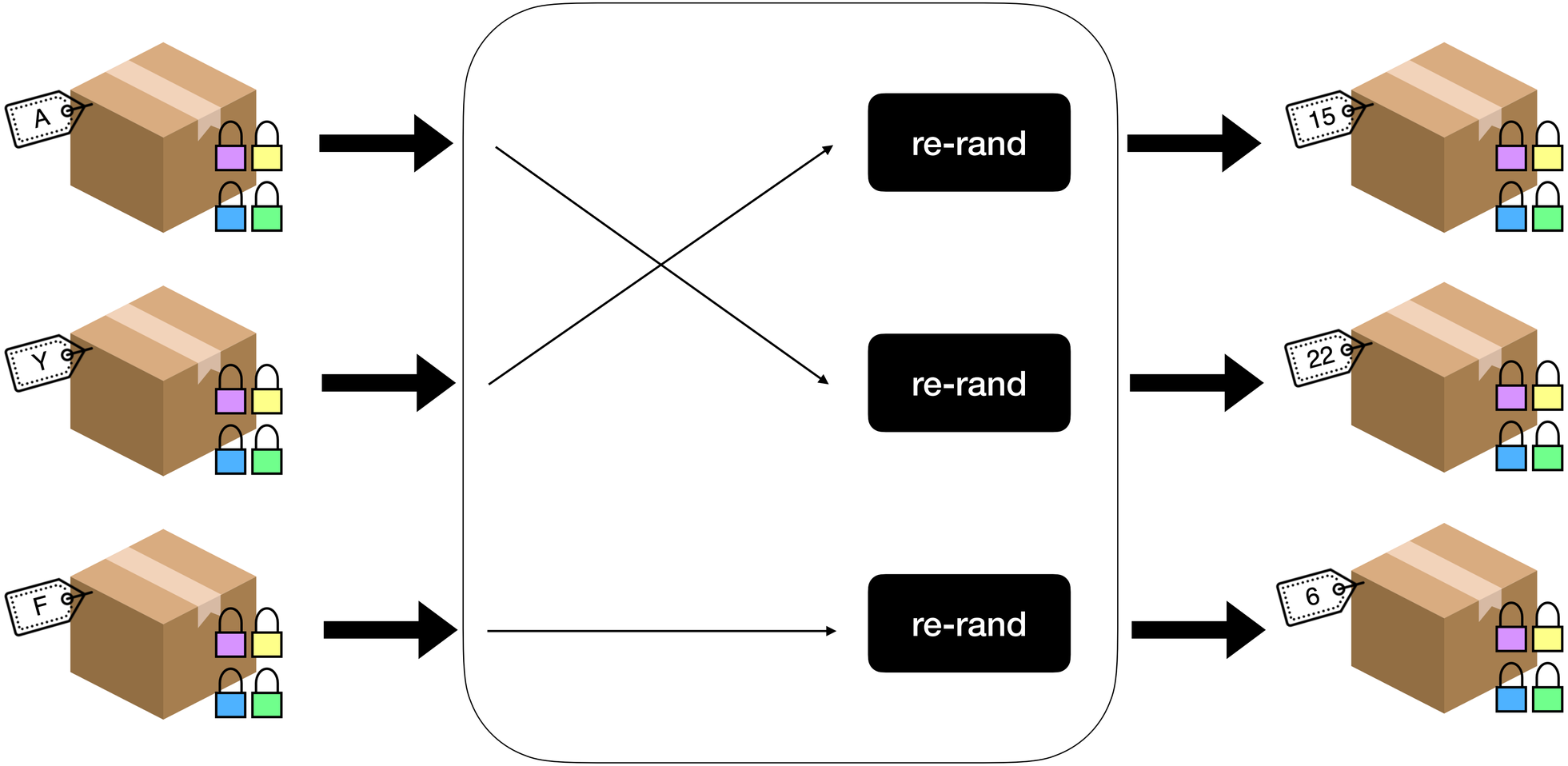
Ciphertext

In reality ciphertexts are unique! Even if you can't decrypt it, you can keep track: the A♣ is in box "A"

Card Lifecycle: Digital Card

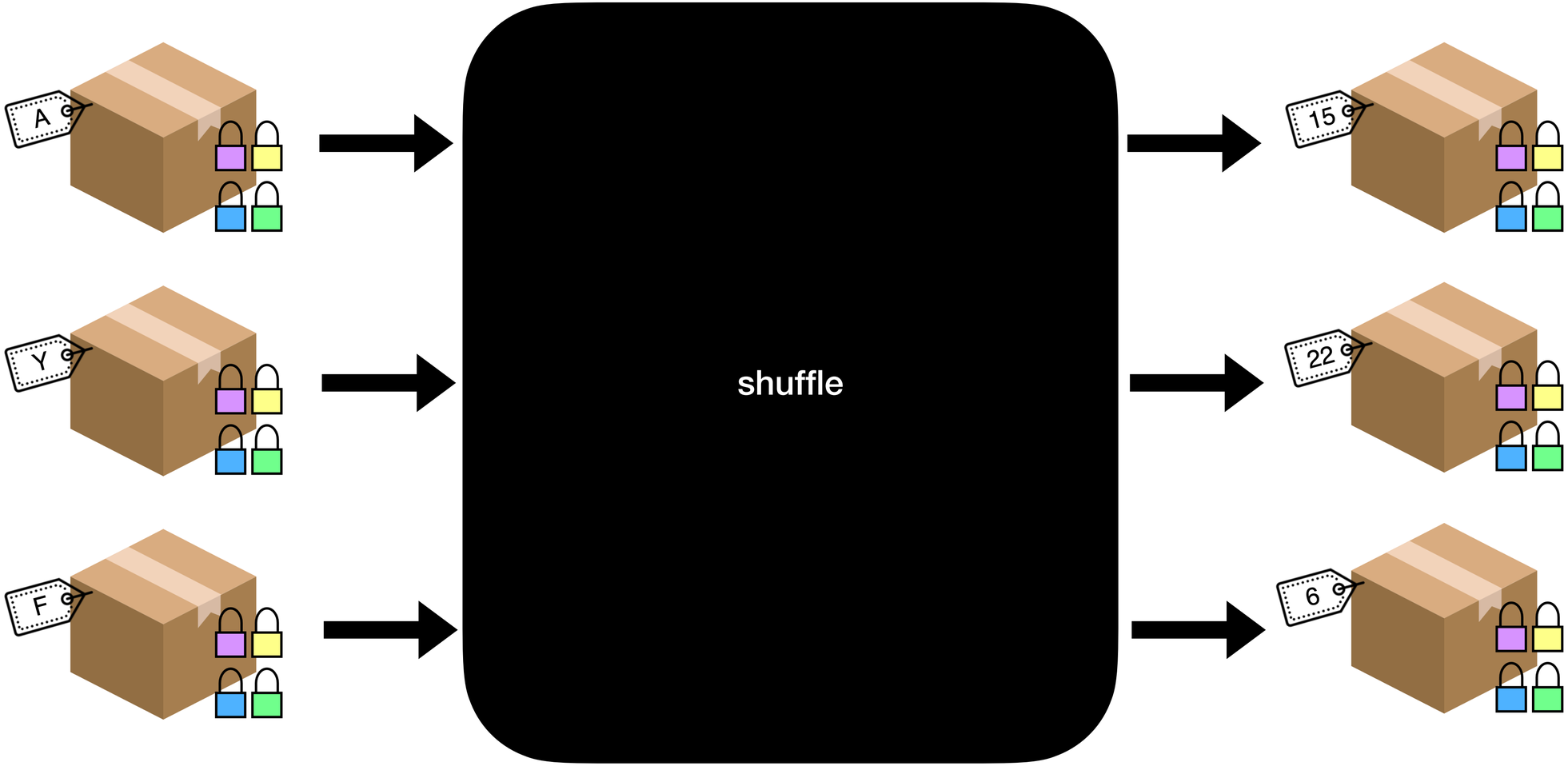


Shuffling the Deck: Permute and Randomise



Shuffling the Deck: Private Shuffle

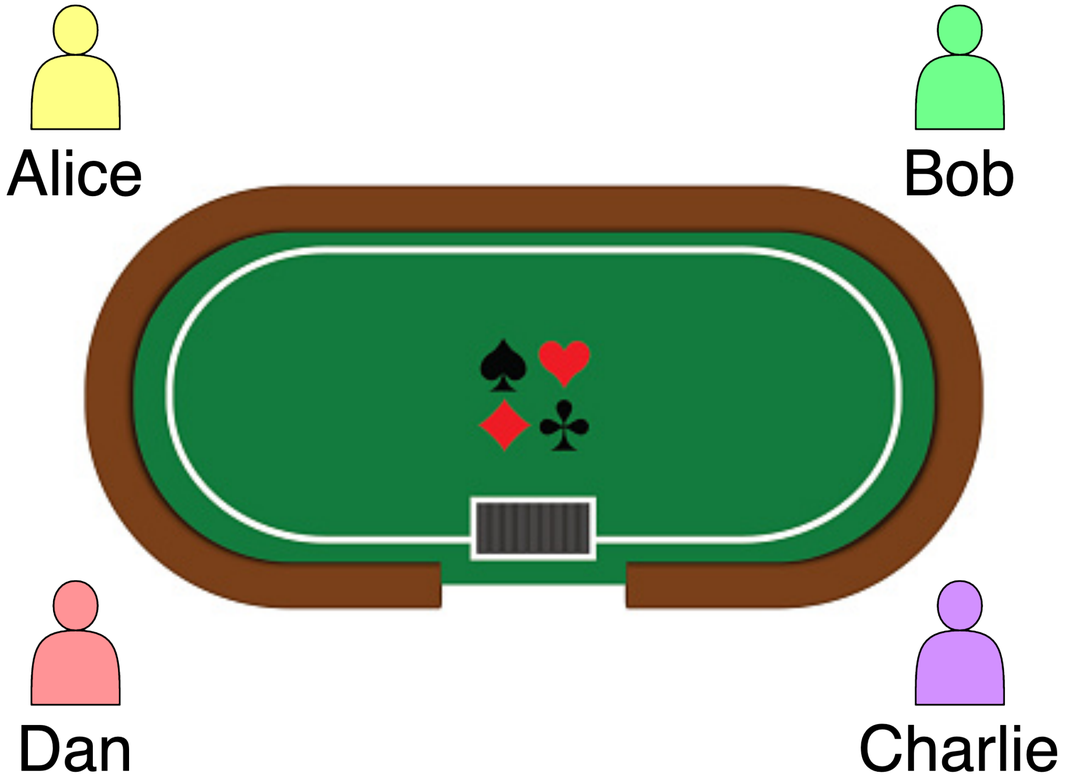
Hiding the permutation and the re-randomisation allows for a private shuffle. Only the player performing the shuffle knows how to revert it.



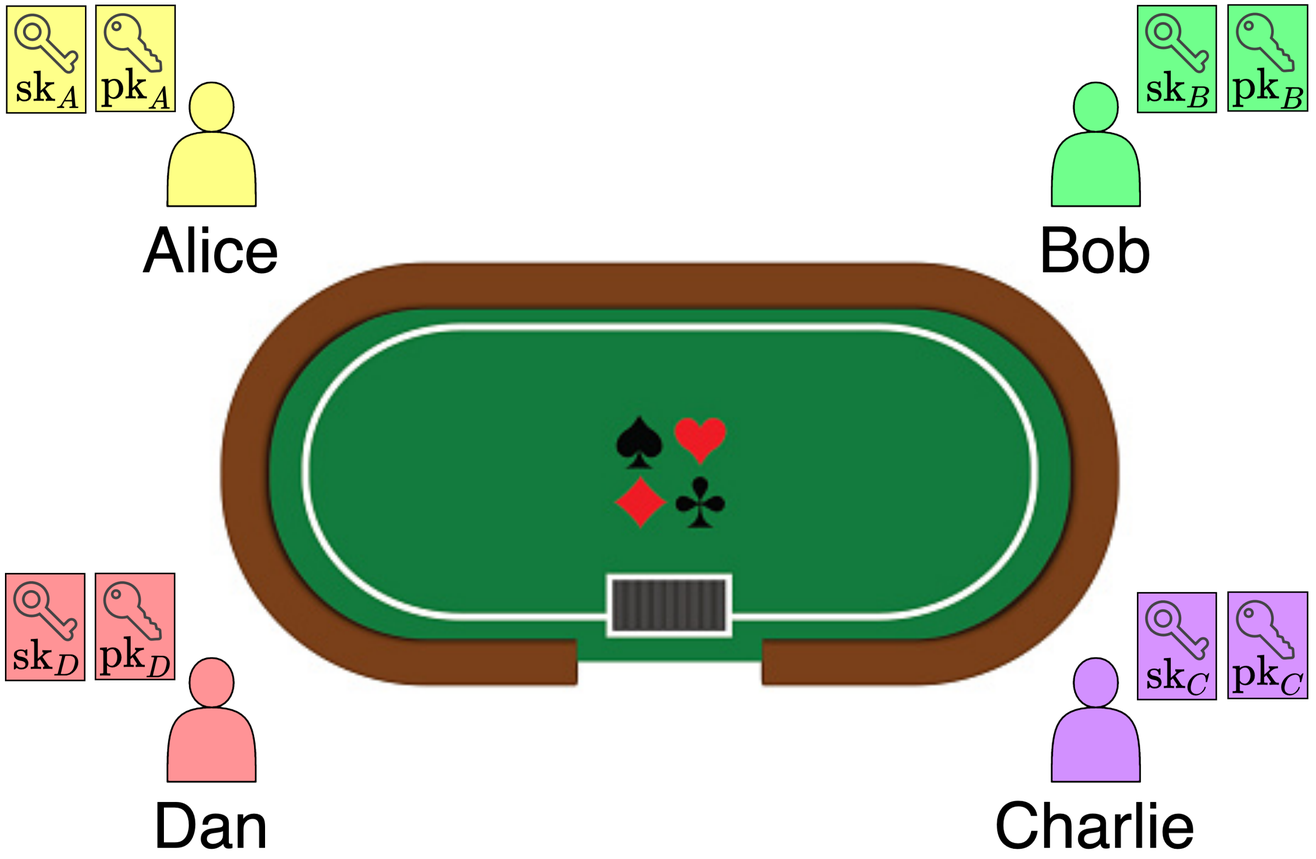
Shuffling the Deck: Verified Shuffle

- Need to prove that the output deck is a permutation and re-randomisation of the input deck, i.e. no cards added or dropped
- Fixed number of cards: Texas Hold'Em uses 52, French Tarot uses 78, Uno uses 108
    - Asymptotics don't really matter
- We use the Bayer-Groth shuffle argument:
    - Works in any setting where discrete log is hard (no pairings or FFTs required)
    - Not succinct… But not a big deal given the number of cards!

Round of Poker: Setup



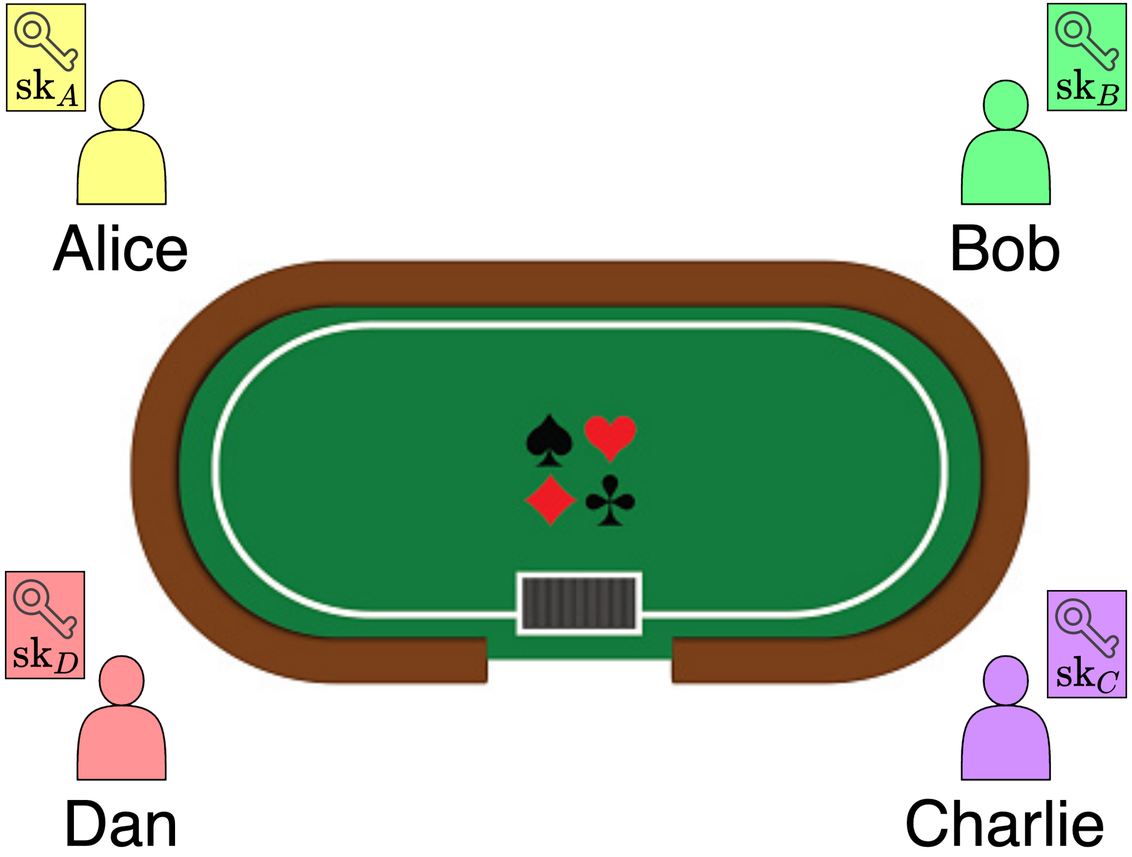
Round of Poker: Setup



Round of Poker: Setup

Verify and aggregate public keys

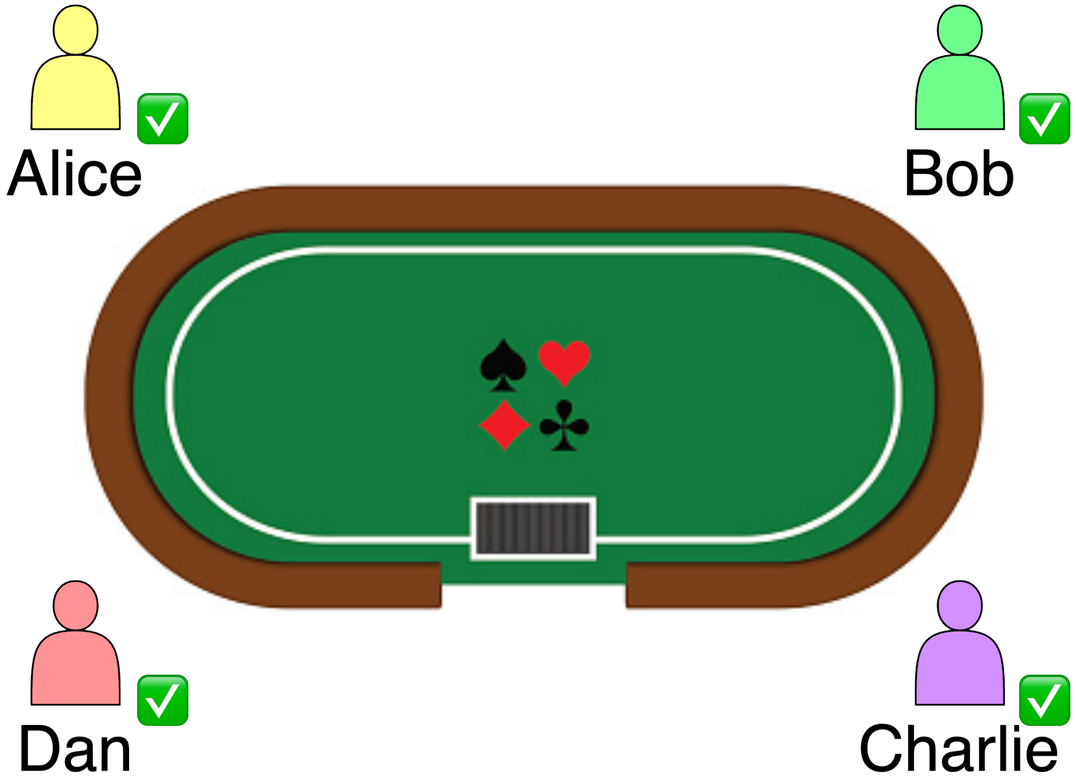
$\mathrm{pk}_A + \mathrm{pk}_B + \mathrm{pk}_C + \mathrm{pk}_D = \mathrm{pk}_{Agg}$



Round of Poker: Setup

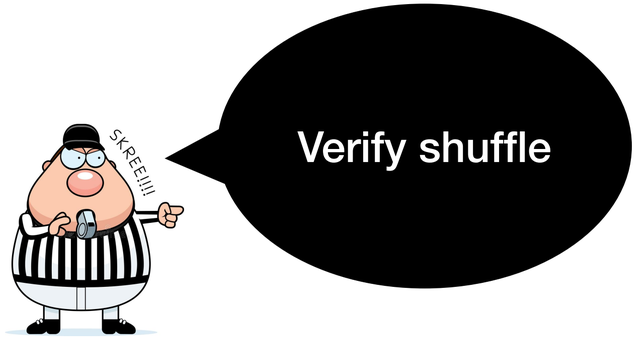
Bob's view

	Card Value	Ciphertext
1	A♥	0x8633E124
2	2♥	0x9734A149
3	3♥	0x249BF587
…	…	…
51	Q♠	0x314F4987
52	K♠	0x87372027



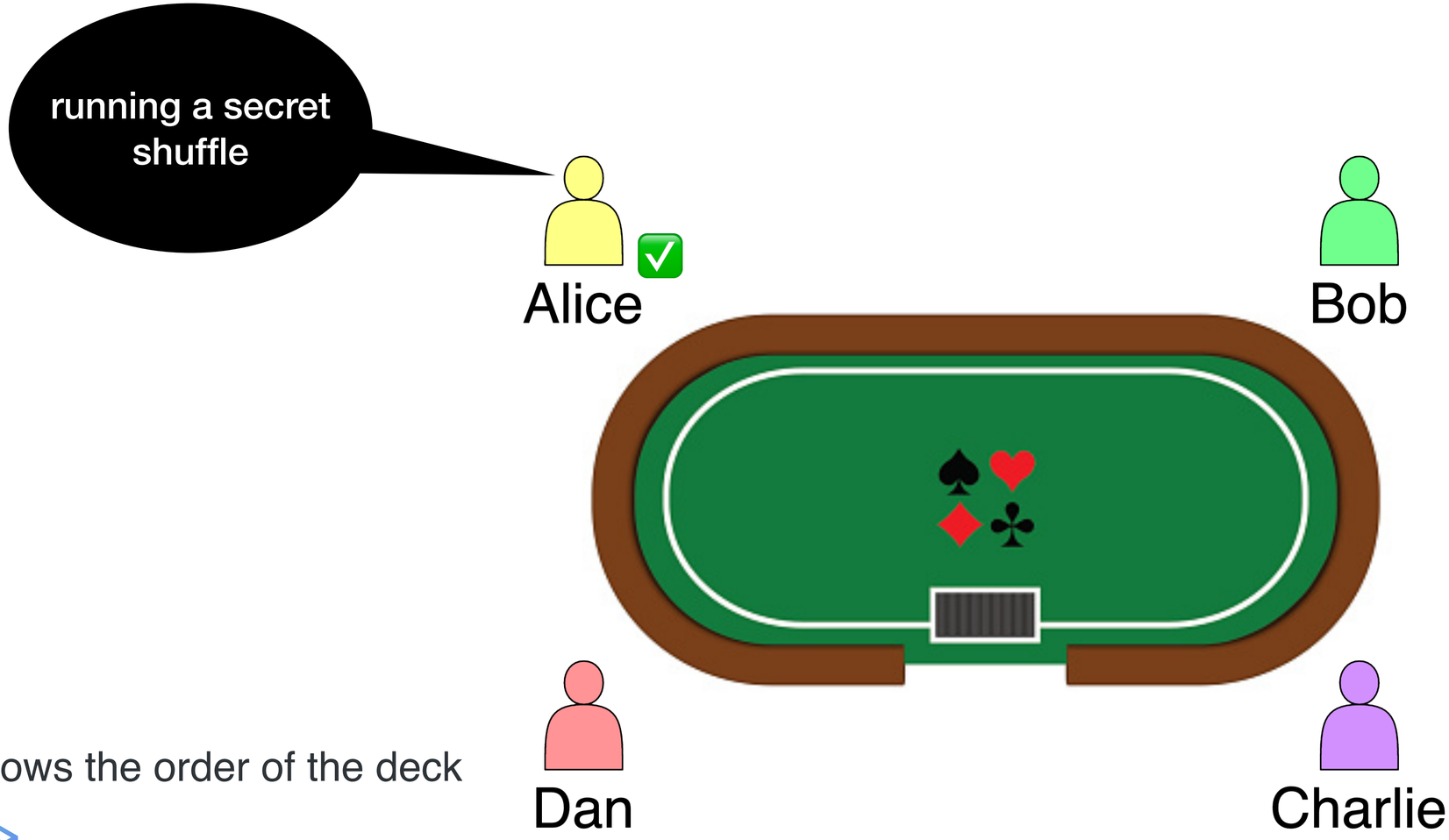
✅ = knows the order of the deck

Round of Poker: Shuffle



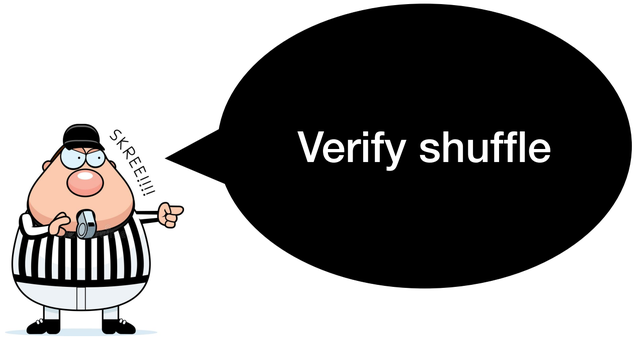
Bob's view

	Card Value	Ciphertext
1	???	0x73683435
2	???	0x097A4430
3	???	0x1475F394
…	…	…
51	???	0x13083930
52	???	0x239E38B3



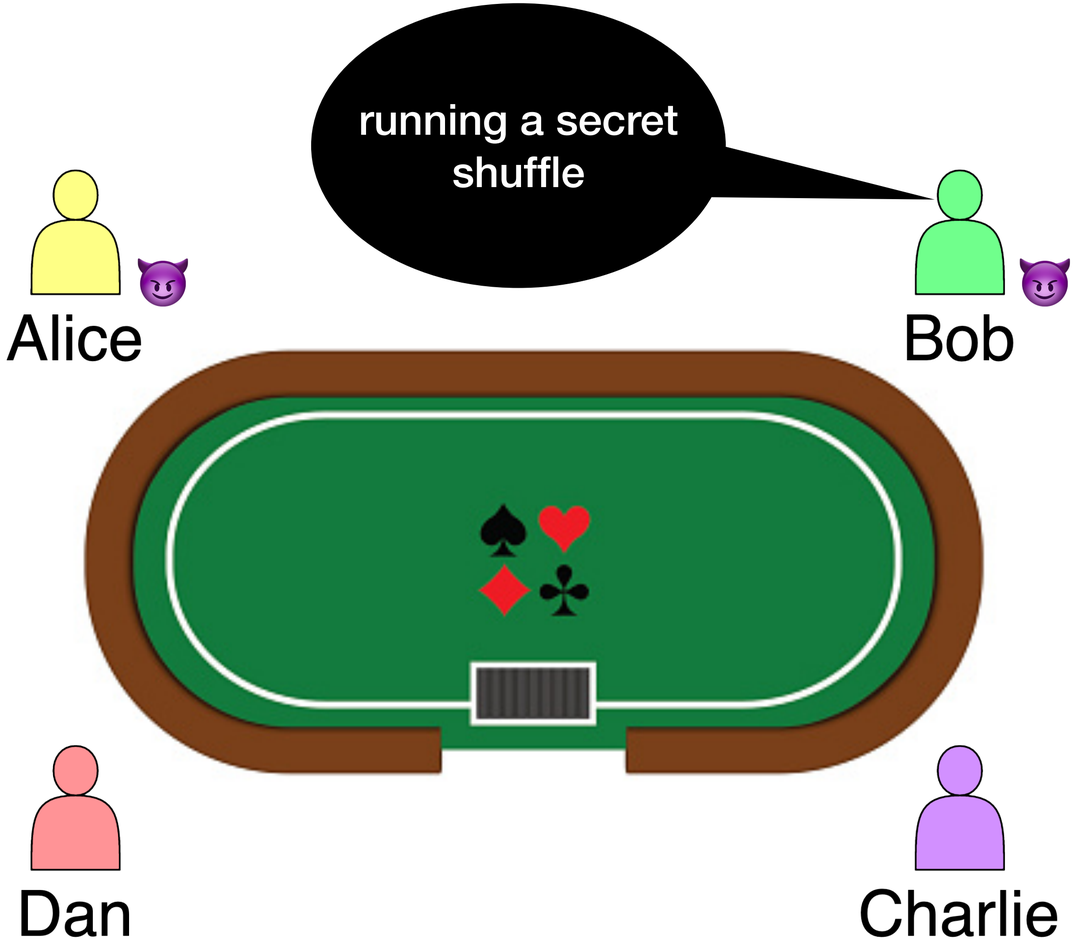
✅ = knows the order of the deck

Round of Poker: Shuffle



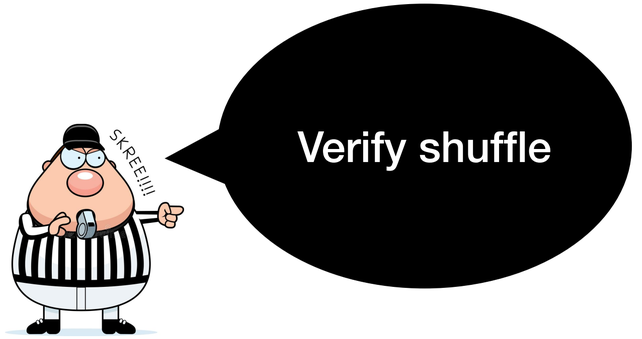
Bob's view

	Card Value	Ciphertext
1	???	0xE3B56449
2	???	0x0976F30A
3	???	0x982BC374
…	…	…
51	???	0x684F5932
52	???	0x147870C2



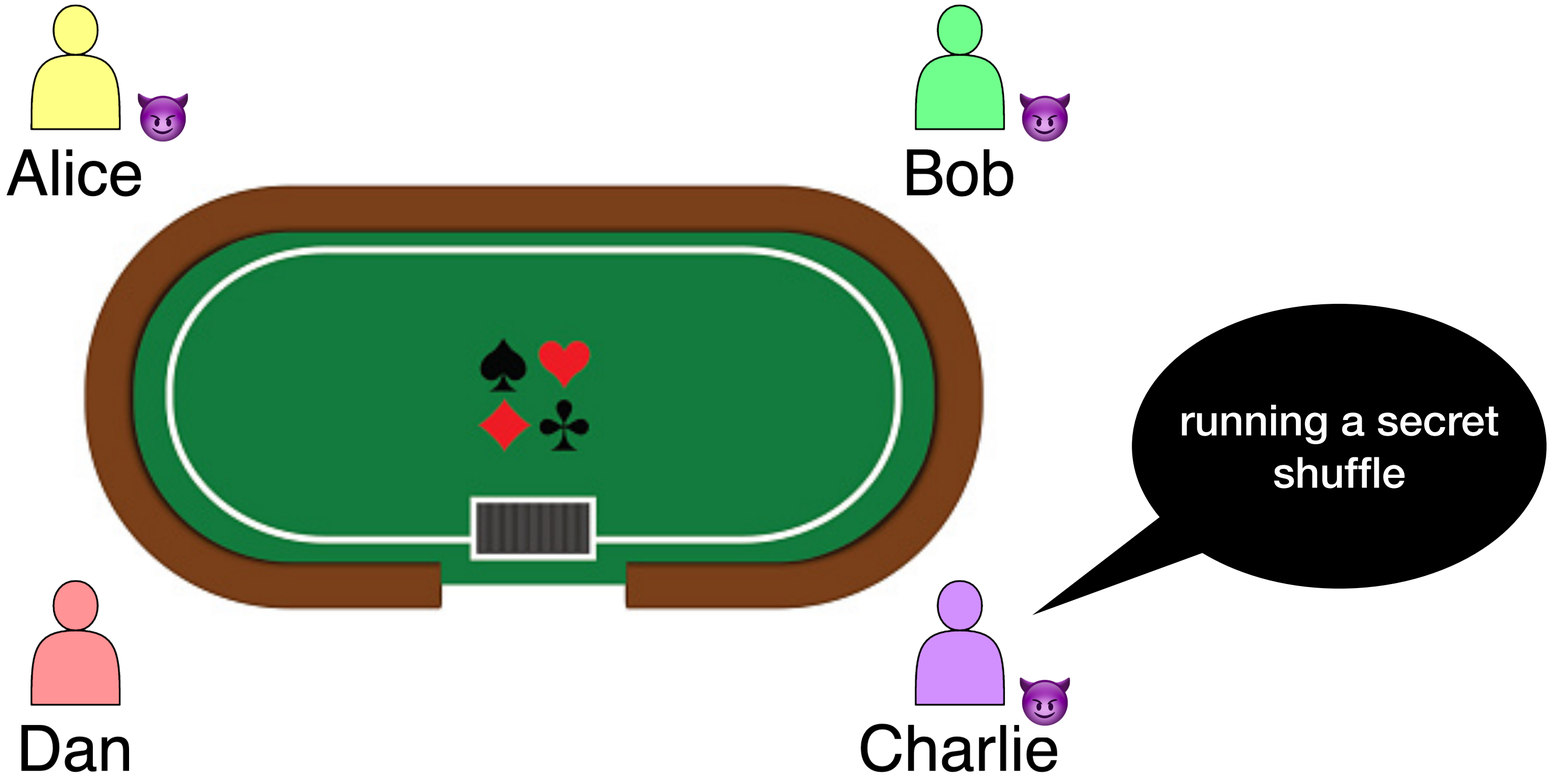
😈 = can collaborate to learn order of the deck

Round of Poker: Shuffle



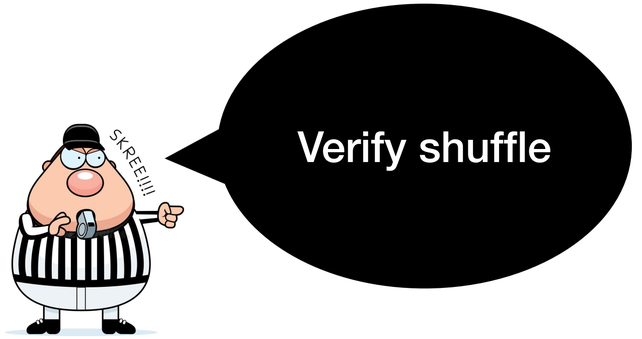
Bob's view

	Card Value	Ciphertext
1	???	0x097F4345
2	???	0x426BA4A5
3	???	0x7B835409
…	…	…
51	???	0x670BB261
52	???	0x0498EA92



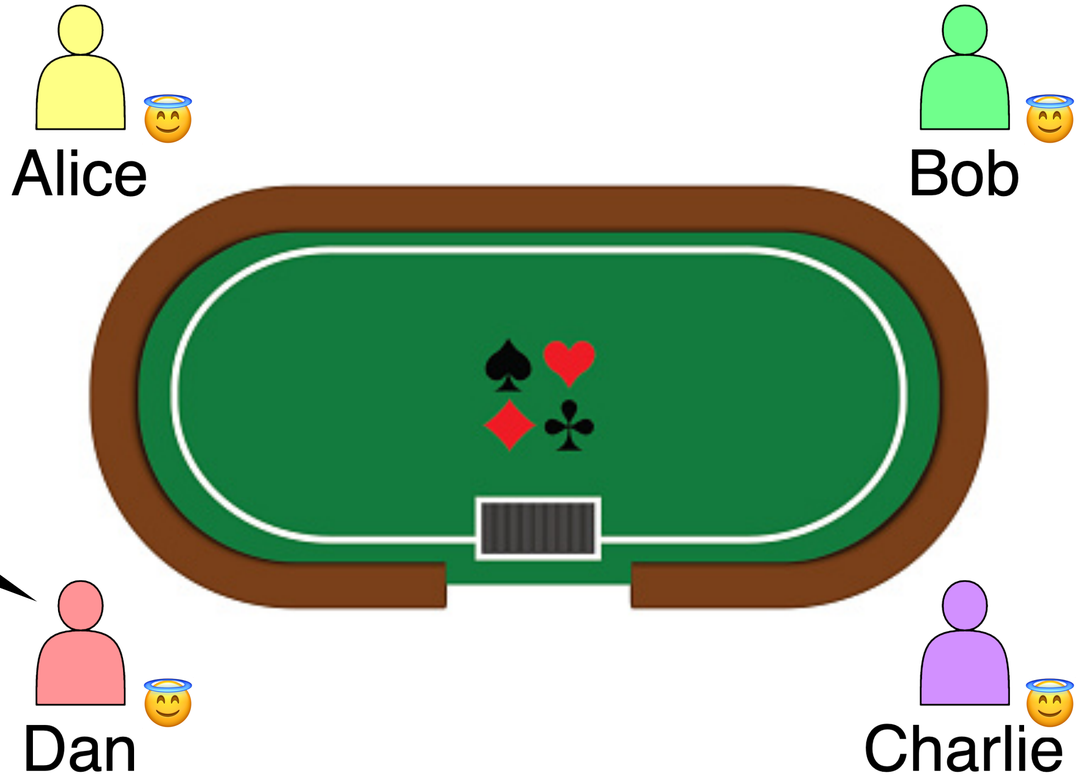
😈 = can collaborate to learn order of the deck

Round of Poker: Shuffle



Bob's view

	Card Value	Ciphertext
1	???	0x920B3482
2	???	0x8A74E126
3	???	0x147D8224
…	…	…
51	???	0x92767035
52	???	0x547FEB49

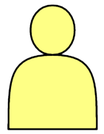


😇 = must all collaborate to learn card values. Play fair!

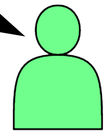
Round of Poker: Give 1st card to Alice

Bob's view

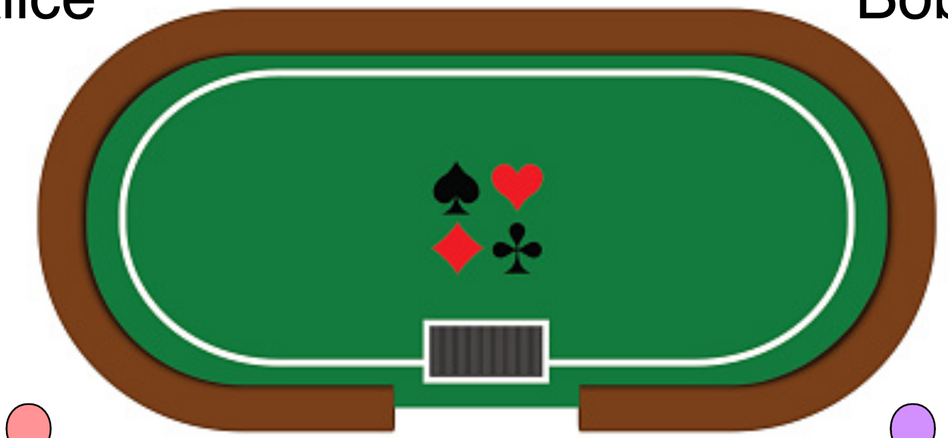
	Card Value	Ciphertext
1	???	0x920B3482
2	???	0x8A74E126
3	???	0x147D8224
…	…	…
51	???	0x92767035
52	???	0x547FEB49

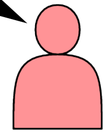


Alice

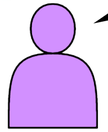


Bob





Dan



Charlie

Round of Poker: Give 2nd card to Bob

Verify reveals

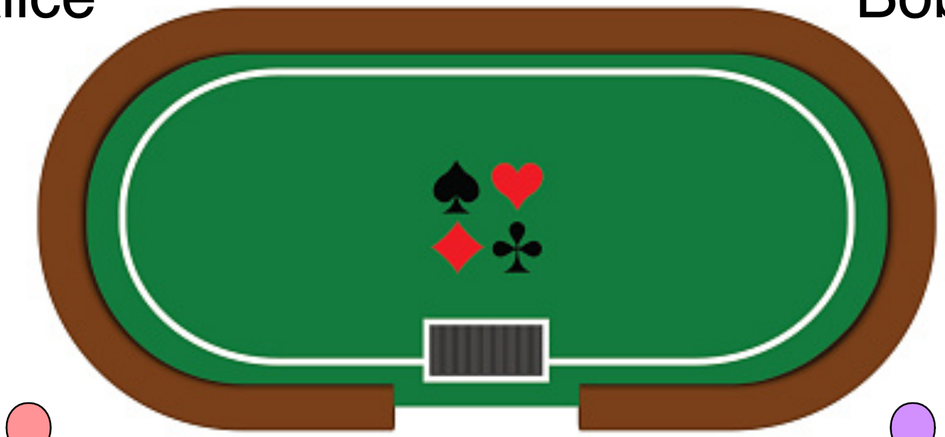
Bob's view

	Card Value	Ciphertext
1	???	0x920B3482
2	5♣	0x8A74E126
3	???	0x147D8224
…	…	…
51	???	0x92767035
52	???	0x547FEB49

running reveal for card 2

Alice

Bob



running reveal for card 2

Dan

running reveal for card 2

Charlie

Round of Poker: Reveal 10th Card (Flop)

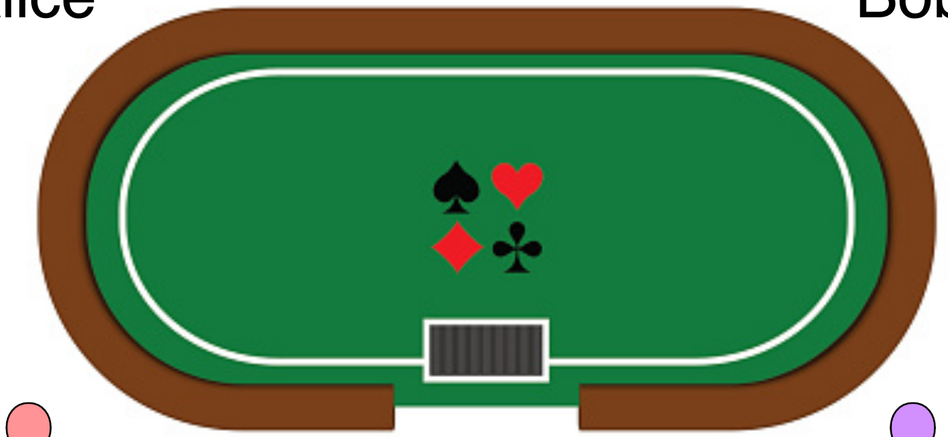
Verify reveals

running reveal for card 10

Alice

running reveal for card 10

Bob



running reveal for card 10

Dan

running reveal for card 10

Charlie

Bob's view

	Card Value	Ciphertext
1	???	0x920B3482
2	5♣	0x8A74E126
3	???	0x147D8224
…	…	…
10	K♦	0x5C43A887
…	…	…
51	???	0x92767035
52	???	0x547FEB49

Math (1): n-out-of-n Threshold El-Gamal Encryption

- Private keys: x_i chosen at random in $\mathbb{Z}_q$ (for prime q)
- Public keys: $H_i = x_i G$ (for a generator G)
- Key Aggregation: $H_{agg} = H_1 + H_2 + \ldots + H_n$
- Encryption:

$$\mathscr{E}_H(M, \alpha): \quad (C_a, C_b) \leftarrow (\alpha G, M + \alpha H)$$

- Re-randomise:

$$\mathscr{E}_H'((C_a, C_b), \beta): \quad (C_a', C_b') \leftarrow (C_a + \beta G, C_b + \beta H)$$

Important:

$$C_a' = (\alpha + \beta)G$$
$$C_b' = M + (\alpha + \beta)H$$

- Decryption:

player i publishes $D_i \leftarrow x_i C_a$. Compute $M = C_b - (D_1 + D_2 + \ldots + D_n)$

Math (1): n-out-of-n Threshold El-Gamal Encryption

Verifiable

- Private keys: x_i chosen at random in $\mathbb{Z}_q$ (for prime q)
- Public keys: $H_i = x_i G$ (for a generator G)

Schnorr proof of knowledge of discrete log

- Key Aggregation: $H_{agg} = H_1 + H_2 + \ldots + H_n$
- Encryption:

$$\mathscr{E}_H(M, \alpha): \quad (C_a, C_b) \leftarrow (\alpha G, M + \alpha H)$$

Chaum-Pedersen proof that $log_G(C_a) = log_H(M - C_b)$

- Re-randomise:

$$\mathscr{E}_H'((C_a, C_b), \beta): \quad (C_a', C_b') \leftarrow (C_a + \beta G, C_b + \beta H)$$

Chaum-Pedersen proof that $log_G(C_a' - C_a) = log_H(C_b' - C_b)$

- Decryption:

player i publishes $D_i \leftarrow x_i C_a$. Compute $M = C_b - (D_1 + D_2 + \ldots + D_n)$

Chaum-Pedersen proof that $log_{C_a}(D_i) = log_G(H_i)$

Math (2): Bayer-Groth Shuffle Argument

Protocol overview:

1. P commits to a permutation of the vector $[1,2,3,\ldots,N]$
2. Verifier sends a challenge x
3. P commits to a permutation of the vector $[x, x^2, x^3, \ldots, x^N]$
4. Verifier sends challenges y and z
5. P and V run a *product argument* to convince V that the same permutation was used in steps 1 and 3.
6. Using the commitment from step 3, P and V run a *multi-exponentiation argument* to convince V that the shuffle is valid and uses the permutation from step 1.

Implementation

- Open-source library, available on Github https://github.com/geometryresearch/mental-poker
- Features:
  - written in Rust using Arkworks
  - modular design
  - curve-agnostic
  - compiles to WebAssembly
- Performance: prove a shuffle in 50ms, verification in 0.6ms for the StarkNet finite field on a consumer laptop
- Help us! Looking to translate the verifier into your favourite DSL

Conclusion

- We've seen how to use **threshold encryption** and **zero knowledge proofs** to play a **fair, decentralised game of poker**
- **Open-source** library ready to go
- Next steps with your help: write the shuffle verifier in a smart contract DSL, write a **game**!

Resources

- A. Shamir, R. Rivest, and L. Adleman, "Mental Poker", Technical Memo LCS/TM-125, Massachusetts Institute of Technology, April 1979. https://apps.dtic.mil/dtic/tr/fulltext/u2/a066331.pdf
- Barnett, Adam, and Nigel P. Smart. "Mental poker revisited." In IMA International Conference on Cryptography and Coding, pp. 370-383. Springer, Berlin, Heidelberg, 2003.
- Bayer, Stephanie, and Jens Groth. "Efficient zero-knowledge argument for correctness of a shuffle." In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 263-280. Springer, Berlin, Heidelberg, 2012

hello@geometry.xyz

@_geometry_

LONDON ST. HELIER LISBON BERLIN BELGRADE TEL AVIV