Cryptography for Blockchains beyond ECDSA and SHA256

Signatures and Zero Knowledge Proofs

Benedikt Bünz
Stanford University

Overview

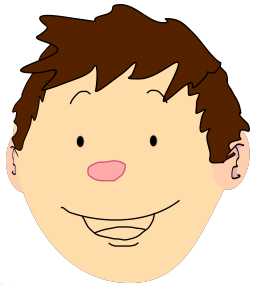
1. Signatures
    1. ECDSA
    2. BLS
    3. Threshold Signatures
    4. Ring Signatures
    5. Blind Signatures
2. Zero-Knowledge Proofs
    1. An illustrative example (SUDOKU)
    2. Sigma protocols
    3. SNARKs
    4. PCPs, CS-Proofs and STARKs
    5. Bulletproofs

Signatures

Did Bob or Bart say this?

Alice

Send 1 BTC to Alice



Bob

Bart

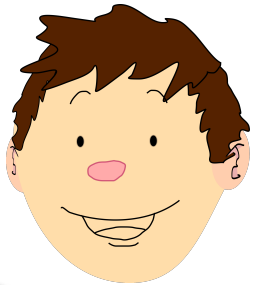
Signatures

Did Bob or Bart say this?

Alice

Send 1 BTC to Alice

Bob



Bob

Bart

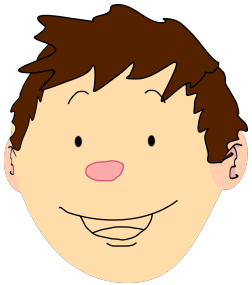
Signature (Formal Definiton)

- Keygen→ (sk,pk)
- Sign (sk,m) → σ
- Verify(pk, σ,m) →{0,1}

- Correctness: Verify(PK,SIGN(SK,M),M)=1

- SECURITY: After seeing n signatures no adversary can create a signature on new message

Signature (Diagram)



(sk,pk)←Keygen

pk

σ ←Sign(sk,m)

m, σ

Verify(pk, m, σ)

Signature (ECDSA)

- Used in Bitcoin (and other Cryptocurrencies)
- Designed because of a patent conflict
- Malleable: Given (pk,σ,m)-> σ' with Verify(pk,σ',m)
  - -> Transaction malleability
  - -> Fooled Mt. Gox Cash Out Twice
  - After seeing n signatures no adversary can create a new signature on any message

Preliminaries: Discrete Log and Pairings

- Random $x \in \mathbb{Z}_p^*$
- Given $g, g^x \in \mathbb{G}$ it's hard to produce x
- Pairing (Bilinear Map) $e: \mathbb{G}, -> \mathbb{G}_T$, $\mathbb{G}$ is a special elliptic curve
  - $g^a, g^b \in \mathbb{G}, e(g^a, g^b) = e(g, g)^{a*b}$
  - $e(g^a, h) = e(g, h^a) = e(g, h)^a$
  - $e(g, u * v) = e(g, u) * e(g, v)$

BLS: Signatures

- Eliptic curve $\mathbb{G}$ with pairing e and generator g
- H is a hash function that hashes into $\mathbb{G}$
- Setup: $x \leftarrow^r \mathbb{Z}_p$. sk: x, pk: g, g^x
- Sign(x,m): $\sigma = H(m)^x$
- Verify(g, g^x, σ,m):
  - $e(\sigma, g) =^?$

BLS: Signatures

- Eliptic curve $\mathbb{G}$ with pairing e and generator g
- H is a hash function that hashes into $\mathbb{G}$
- Setup: $x \leftarrow^r \mathbb{Z}_p$. sk: x, pk: g, g^x
- Sign(x,m): $\sigma = H(m)^x$
- Verify(g, g^x, σ,m):
  - $e(\sigma, g) =^? e(H(m), g^x)$
  - $e(\sigma, g)$

BLS: Signatures

- Eliptic curve $\mathbb{G}$ with pairing e and generator g
- H is a hash function that hashes into $\mathbb{G}$
- Setup: $x \leftarrow^r \mathbb{Z}_p$. sk: x, pk: g, g^x
- Sign(x,m): $\sigma = H(m)^x$
- Verify(g, g^x, σ,m):
  - $e(\sigma, g) =^? e(H(m), g^x)$
  - $e(\sigma, g) = e(H(m)^x, g)$

BLS: Signatures

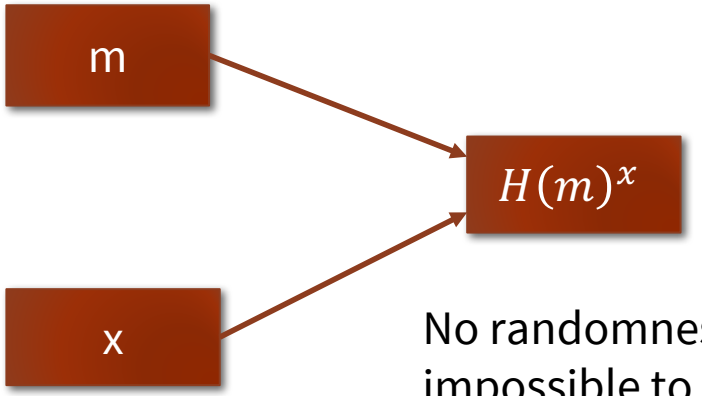
- Eliptic curve $\mathbb{G}$ with pairing e and generator g
- H is a hash function that hashes into $\mathbb{G}$
- Setup: $x \leftarrow^r \mathbb{Z}_p$. sk: x, pk: g, g^x
- Sign(x,m): $\sigma = H(m)^x$
- Verify(g, g^x, σ,m):
  - $e(\sigma, g) =^? e(H(m), g^x)$
  - $e(\sigma, g) = e(H(m)^x, g) = e(H(m), g)^x$

BLS: Signatures

- Eliptic curve $\mathbb{G}$ with pairing e and generator g
- H is a hash function that hashes into $\mathbb{G}$
- Setup: $x \leftarrow^r \mathbb{Z}_p$. sk: x, pk: g, g^x
- Sign(x,m): $\sigma = H(m)^x$
- Verify(g, g^x, σ,m):
  - $e(\sigma, g) =^? e(H(m), g^x)$
  - $e(\sigma, g) = e(H(m)^x, g) = e(H(m), g)^x = e(H(m), g^x)$

σ is 32 bytes!

BLS Properties: Deterministic



No randomness,
impossible to have two valid signatures
for a message, public key pair

BLS Properties: Signature aggregation

- pk_1, m_1, σ_1 and pk_2, m_2, σ_2
- Aggregate signature $\sigma = \sigma_1 * \sigma_2$
- Verify$(g, pk_1, pk_2, m_1, m_2, \sigma)$:
  - $e(\sigma, g) =$

$$e(\sigma, g) =^{?} e(H(m), g^{x})$$

BLS Properties: Signature aggregation

- pk_1, m_1, σ_1 and pk_2, m_2, σ_2
- Aggregate signature $\sigma = \sigma_1 * \sigma_2$
- Verify$(g, pk_1, pk_2, m_1, m_2, \sigma)$:
  - $e(\sigma, g) = e(H(m_1), pk_1)e(H(m_2), pk_2)$
  - $e(\sigma, g) = e(\sigma_1\sigma_2, g)$

$$e(\sigma, g) =^{?} e(H(m), g^x)$$

BLS Properties: Signature aggregation

$$e(\sigma, g) =^{?} e(H(m), g^{x})$$

- pk_1, m_1, σ_1 and pk_2, m_2, σ_2
- Aggregate signature $\sigma = \sigma_1 * \sigma_2$
- Verify$(g, pk_1, pk_2, m_1, m_2, \sigma)$:
  - $e(\sigma, g) = e(H(m_1), pk_1)e(H(m_2), pk_2)$
  - $e(\sigma, g) = e(\sigma_1\sigma_2, g) = e(\sigma_1, g)e(\sigma_2, g)$
  - $e(\sigma_1, g) = e(H(m_1), pk_1)$
  - $e(\sigma_2, g) = e(H(m_2), pk_2)$
  - $e(\sigma_1, g)e(\sigma_2, g) = e(H(m_1), pk_1)e(H(m_2), pk_2)$

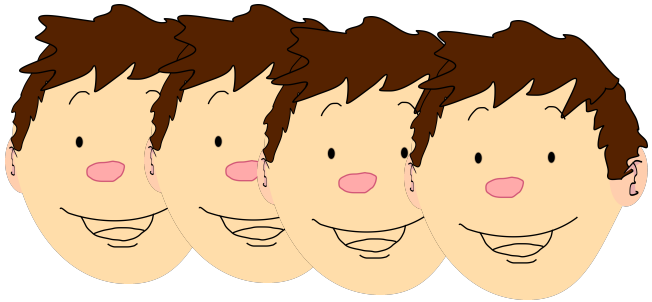
BLS Properties: Signature aggregation

- Take n signatures under n public keys on n messages and create a single small signature
- Each Bitcoin transaction includes public key and is message

- Take all Bitcoin transactions in a block and create a single signature

- Take all Bitcoin transactions in the blockchain and have a single signature
- 32 bytes!

Threshold Signature (Diagram)



Different from multisig (1 pk)

$(sk_1, \ldots, sk_n, pk) \leftarrow \text{Keygen}(t,n)$

pk →

$\sigma \leftarrow \text{Sign}(sk_1, \ldots, sk_t, m)$

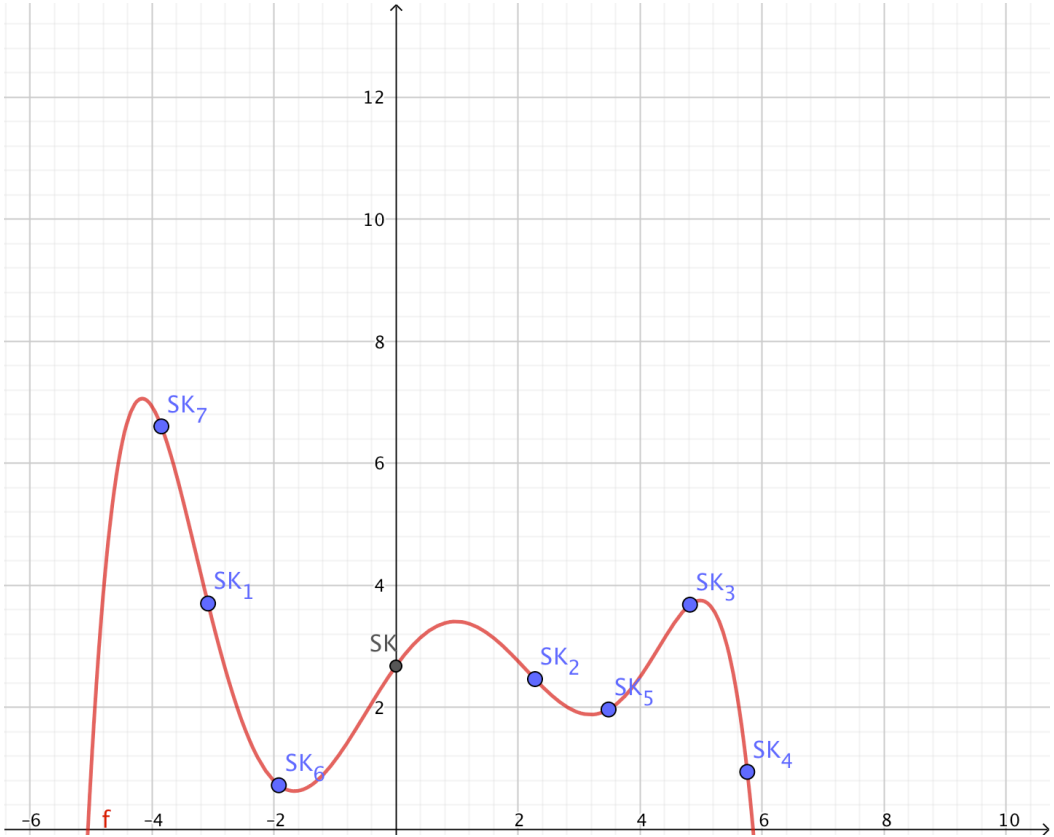
m, σ →

Verify(pk,m, σ)

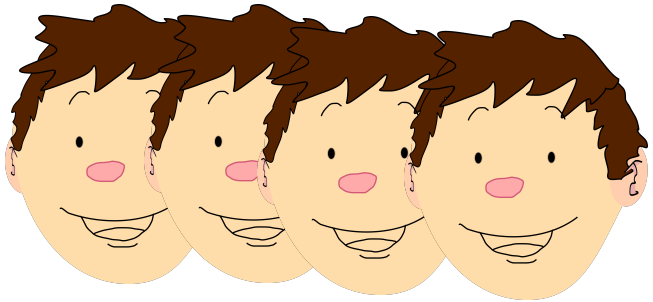
Threshold

- Difficult for ECDSA (Gennaro et al. 16)
- Easy for Schnorr, BLS
- Indistinguishable from normal signature (privacy benefits)
- Can support 1000s of keys/ signatures don’t grow with (t,n)

Threshold signature from Shamir secret sharing



Ring Signature (Diagram)



$(sk_1, \ldots, sk_n, pk) \leftarrow \text{Keygen}(t,n)$

pk →

$\sigma \leftarrow \text{Sign}(sk_i, m)$

m, σ →

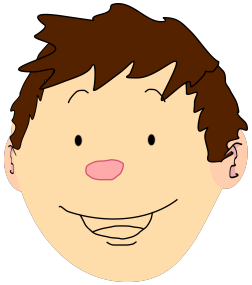
Verify(pk,m, σ)

σ hides i

Ring Signatures

- Used in Monero to hide sender
- Monero's signatures are linear in n
- Can be logarithmic in n (Bootle et al. 2015)

Blind Signature (Diagram)



(sk ,pk)←Keygen(t,n)

pk →

← c

c=Commit(m;r)

b←BlindSign(sk,c)

b →

σ =Unblind(pk,b,r)
s.t. Verify(pk,m, σ)=1

Blind Signature

Separate Custodian from transaction details

Zero Knowledge Proofs of Knowledge

SUDOKUs, SNARKs, STARKs and BULLETs

Zero Knowledge Proof of Knowledge

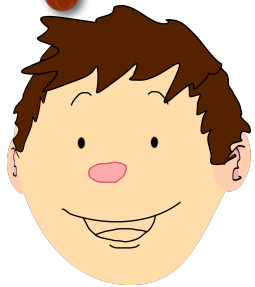
"I know the solution to this complex equation"

"Prove it"

Challenge

Response

No idea what the solution is But Alice must know it



Zero Knowledge Proof of Knowledge Applications

- Confidential Transactions
- Mimblewimble
- ZeroCash
- Hawk
- Zero-Knowledge contingent payments
- Proofs of Solvency for Bitcoin Exchanges
- Confidential Payment Channels
- Blockchain compression
- …

SUDOKU

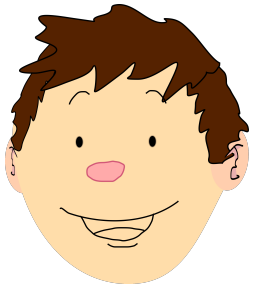
2					1		3	8
								5
	7				6			
							1	3
	9	8	1			2	5	7
3	1					8		
9			8				2	
	5			6	9	7	8	4
4			2	5				

SUDOKU

2	4	9	5	7	1	6	3	8
8	6	1	4	3	2	9	7	5
5	7	3	9	8	6	1	4	2
7	2	5	6	9	8	4	1	3
6	9	8	1	4	3	2	5	7
3	1	4	7	2	5	8	6	9
9	3	7	8	1	4	2	5	6
1	5	2	3	6	9	7	8	4
4	8	6	2	5	7	3	9	1

Zero-Knowledge SUDOKU

2					1		3	8
								5
	7				6			
							1	3
	9	8	1			2	5	7
3	1					8		
9			8				2	
	5			6	9	7	8	4
4				2	5			



”Can you help me with this Sudoku?”

“If you pay me!”

“Prove that you know the solution first”

“Ok”

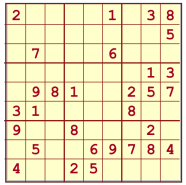
Zero-Knowledge SUDOKU (Gradwohl et al. '05)

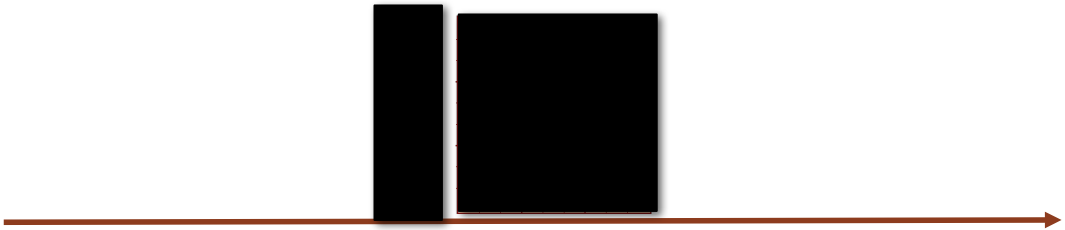
2	4	9	5	7	1	6	3	8
8	6	1	4	3	2	9	7	5
5	7	3	9	8	6	1	4	2
7	2	5	6	9	8	4	1	3
6	9	8	1	4	3	2	5	7
3	1	4	7	2	5	8	6	9
9	3	7	8	1	4	2	5	6
1	5	2	3	6	9	7	8	4
4	8	6	2	5	7	3	9	1

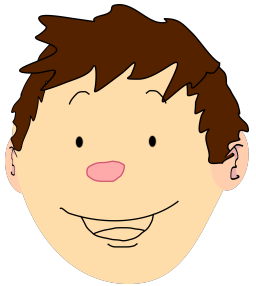
1	5
2	6
3	7
4	1
5	3
6	9
7	4
8	8
9	2

6	1	2	3	4	5	9	7	8
8	9	5	1	7	6	2	4	3
3	4	7	2	8	9	5	1	6
4	6	3	9	2	8	1	5	7
9	2	8	5	1	7	6	3	4
7	5	1	4	6	3	8	9	2
2	7	4	8	5	1	6	3	9
5	3	6	7	9	2	4	8	1
1	8	9	6	3	4	7	2	5

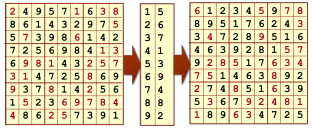
Zero-Knowledge SUDOKU

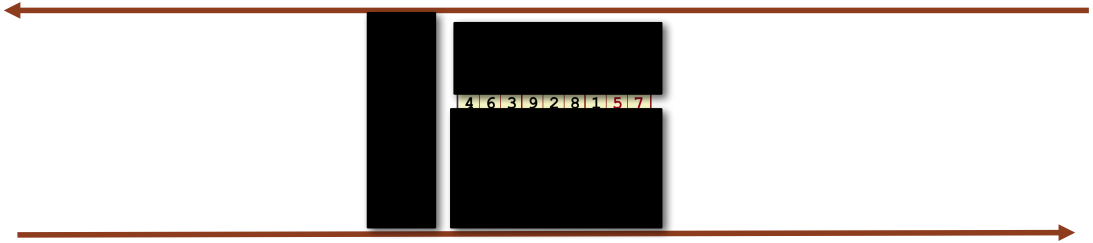




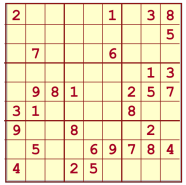


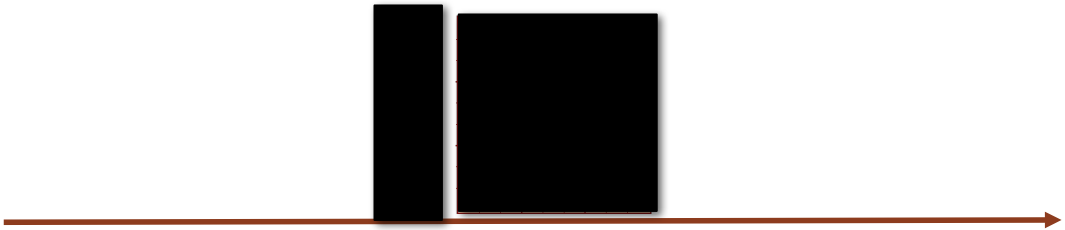
Open Row 4

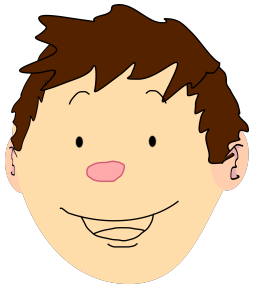




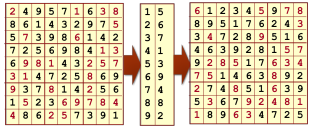
Zero-Knowledge SUDOKU

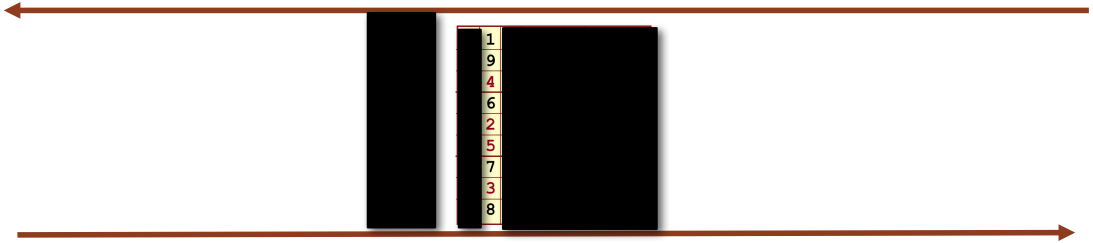






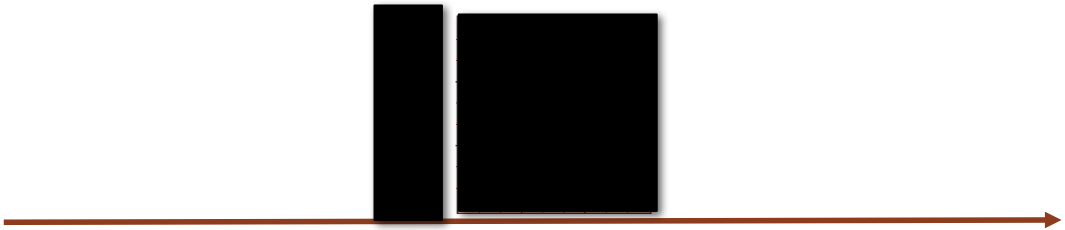
Open Column 2

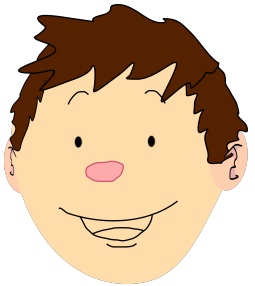




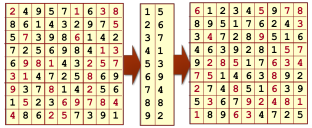
Zero-Knowledge SUDOKU

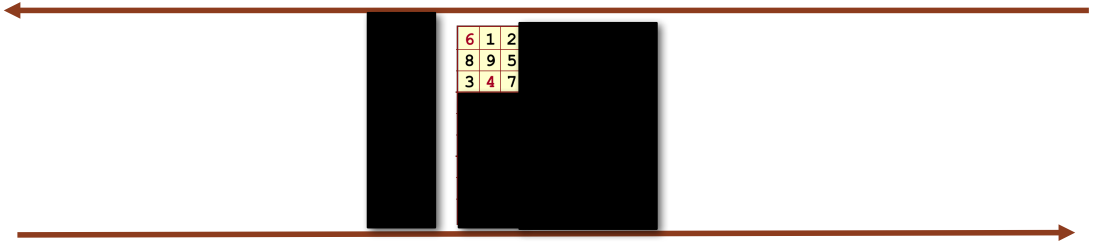
2					1		3	8
								5
	7				6			
							1	3
	9	8	1			2	5	7
3	1					8		
9			8				2	
	5			6	9	7	8	4
4				2	5			





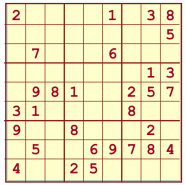
Open Box 1

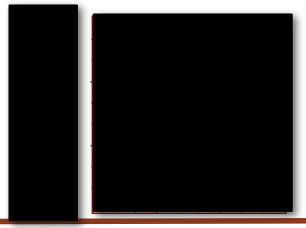


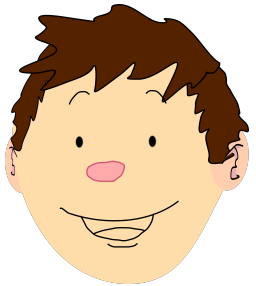


6	1	2
8	9	5
3	4	7

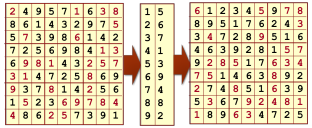
Zero-Knowledge SUDOKU

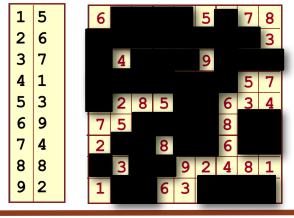






Show original puzzle

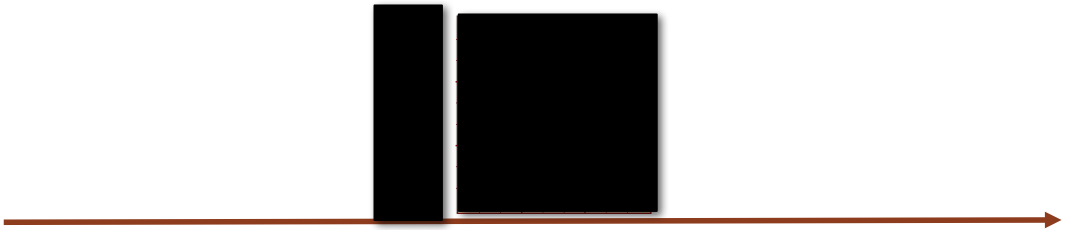


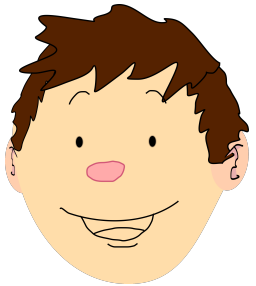


Zero-Knowledge SUDOKU Analysis

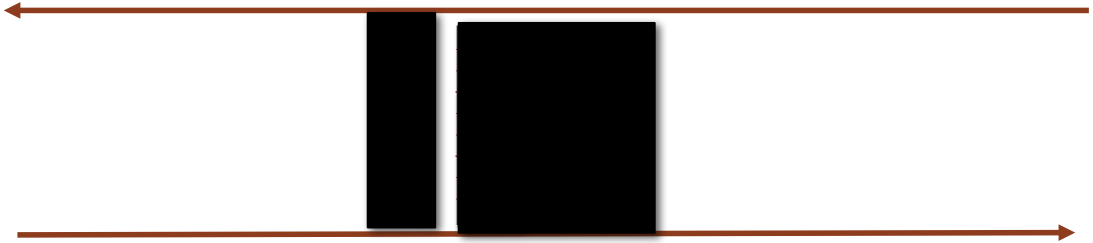
2					1		3	8
								5
	7				6			
							1	3
	9	8	1			2	5	7
3	1					8		
9			8				2	
	5			6	9	7	8	4
4			2	5				

$$P[\text{Cheating}] \leq \frac{27}{28}$$





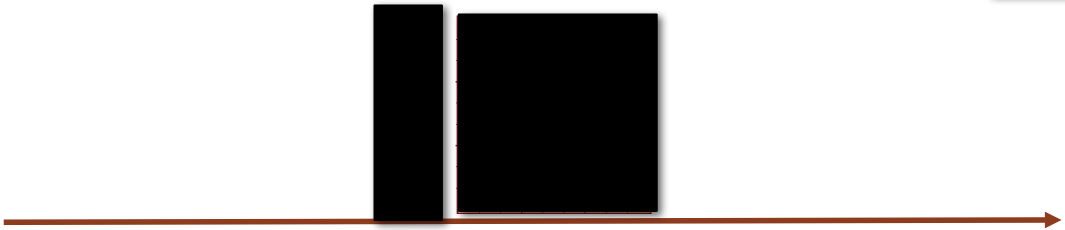
Open Row/Column/Box/Original

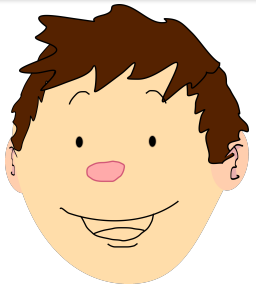


Zero-Knowledge SUDOKU Amplifying

2					1		3	8
								5
	7				6			
							1	3
	9	8	1			2	5	7
3	1					8		
9			8				2	
	5			6	9	7	8	4
4			2	5				

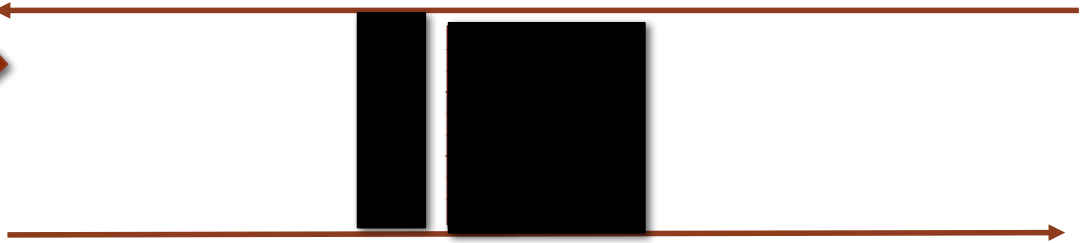
1 try: 0.964
2 tries: 0.929
10 tries: 0.695
100 tries: 0.02
1000 tries: 2^{-52}





Repeat

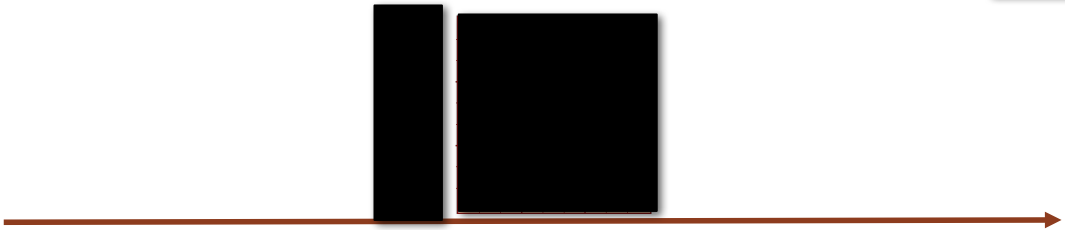
Open Row/Column/Box/Original

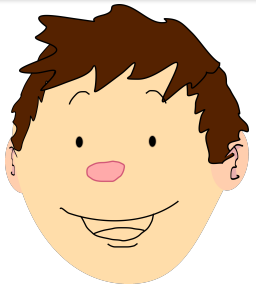


Problem: Bob learns solution

2					1		3	8
								5
		7				6		
							1	3
		9	8	1			2	5	7
3	1						8		
9				8				2	
		5				6	9	7	8	4
4					2	5			

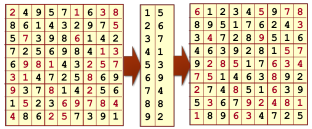
1 try: 0.964
2 tries: 0.929
10 tries: 0.695
100 tries: 0.02
1000 tries: 2^{-52}

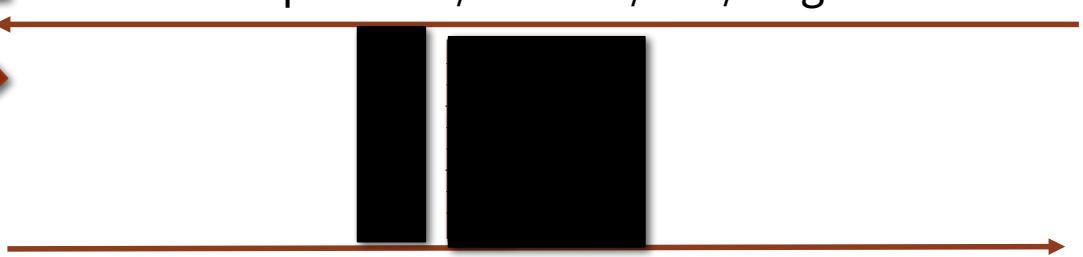




Open Row/Column/Box/Original

Repeat



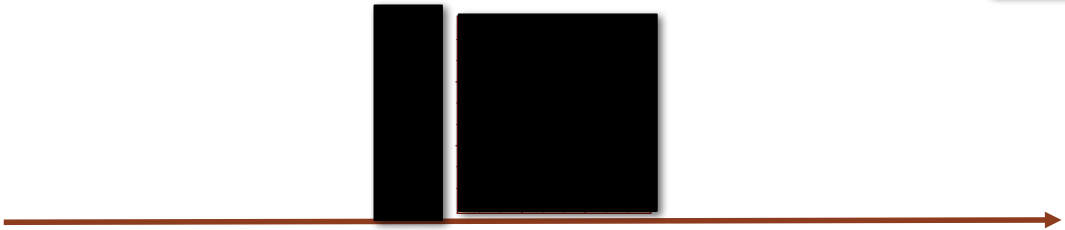


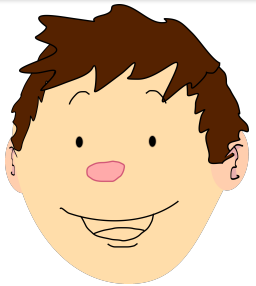
New permutation in every round

Zero-Knowledge

2					1		3	8
								5
	7				6			
							1	3
	9	8	1			2	5	7
3	1					8		
9			8				2	
	5			6	9	7	8	4
4				2	5			

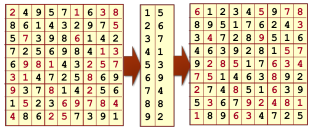
1 try: 0.964
2 tries: 0.929
10 tries: 0.695
100 tries: 0.02
1000 tries: 2^{-52}

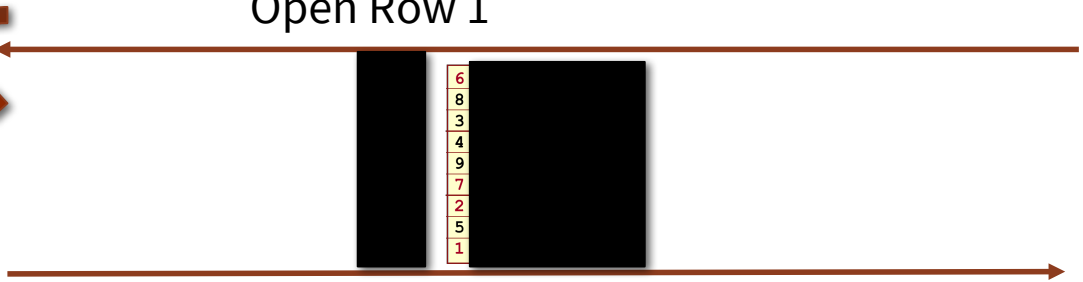




Open Row 1

Repeat



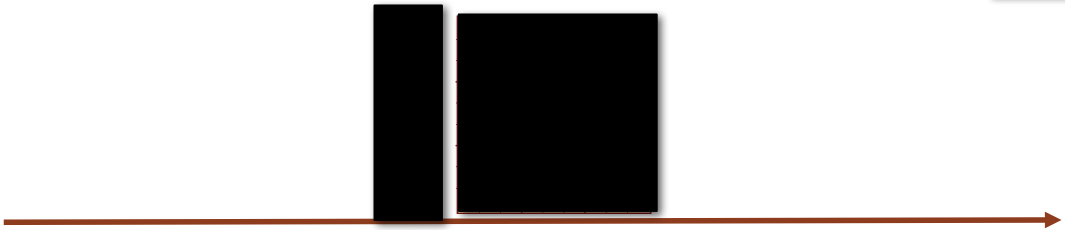


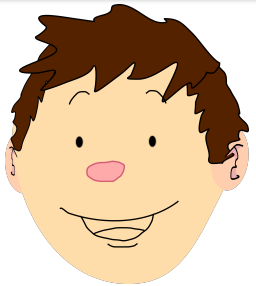
New permutation in every round

Zero-Knowledge

2					1		3	8
								5
	7				6			
							1	3
	9	8	1			2	5	7
3	1					8		
9			8				2	
	5			6	9	7	8	4
4				2	5			

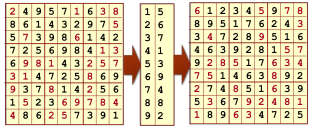
1 try: 0.964
2 tries: 0.929
10 tries: 0.695
100 tries: 0.02
1000 tries: 2^{-52}

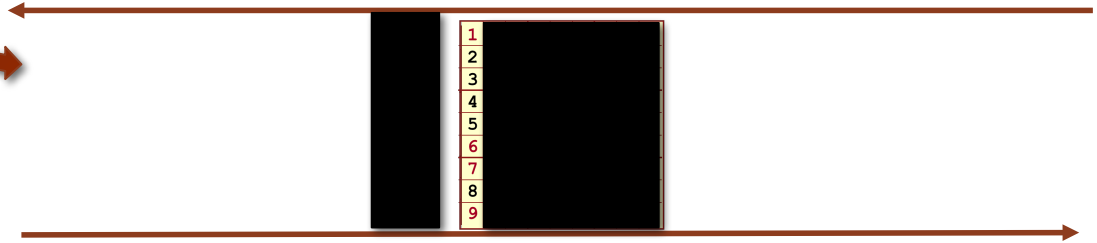




Open Row 1

Repeat



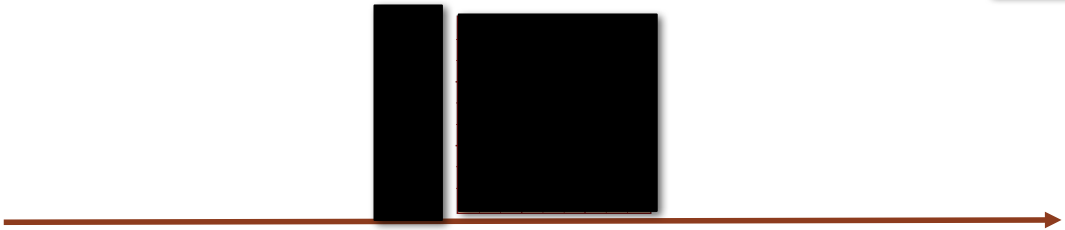


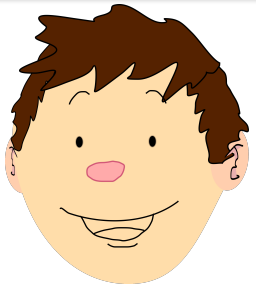
New permutation in every round

Zero-Knowledge

2					1		3	8
								5
	7				6			
							1	3
	9	8	1			2	5	7
3	1					8		
9			8				2	
	5			6	9	7	8	4
4				2	5			

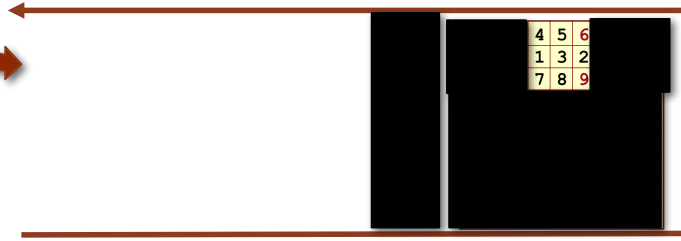
1 try: 0.964
2 tries: 0.929
10 tries: 0.695
100 tries: 0.02
1000 tries: 2^{-52}

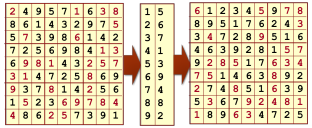




Open Box 2

Repeat

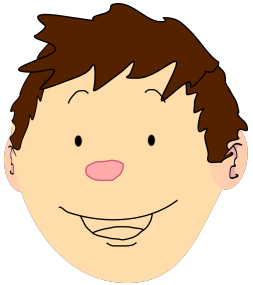




New permutation in every round

Zero-Knowledge for public keys (Sigma protocol)

I know x such that $g^x = y$



$r \leftarrow \mathbb{Z}_p$

$A = g^r$

c

$c \leftarrow \mathbb{Z}_p$

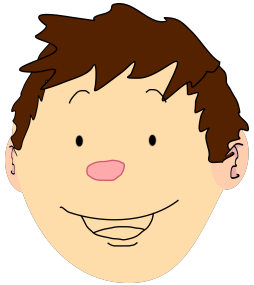
$s = r + c * z$

s

$g^s =^?$

Zero-Knowledge for public keys (Sigma protocol)

I know x such that $g^x = y$



$r \leftarrow \mathbb{Z}_p$

$A = g^r$

c

$c \leftarrow \mathbb{Z}_p$

$s = r + c * z$

s

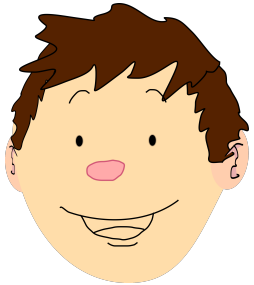
$g^s =^? \mathrm{A} * \mathrm{y}^c$

$\mathrm{A} * \mathrm{y}^c = g^r * g^{x*c}$

$g^r * g^{x*c} = g^{r+x*c}$

Non-Interactive Zero-Knowledge (NIZK)

I know x such that $g^x = y$



$r \leftarrow \mathbb{Z}_p$

$A = g^r$

c

$c \leftarrow \mathbb{Z}_p$

$s = r + c * z$

s

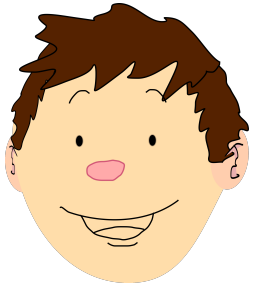
$g^s =^? \mathrm{A} * \mathrm{y}^c$

$\mathrm{A} * \mathrm{y}^c = g^r * g^{x*c}$

$g^r * g^{x*c} = g^{r+x*c}$

Non-Interactive Zero-Knowledge (NIZK)

I know x such that $g^x = y$



$r \leftarrow \mathbb{Z}_p$

$A = g^r$

c

$c \leftarrow \mathbb{Z}_p$

$s = r + c * z$

s

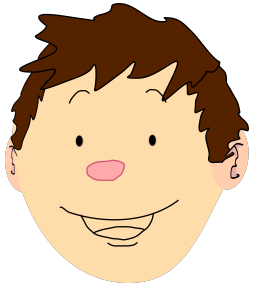
$g^s =^? \mathrm{A} * \mathrm{y}^c$

$\mathrm{A} * \mathrm{y}^c = g^r * g^{x*c}$

$g^r * g^{x*c} = g^{r+x*c}$

Non-Interactive Zero-Knowledge (NIZK)

I know x such that $g^x = y$



$r \leftarrow \mathbb{Z}_p$

$A = g^r$

$c = H(A, y)$

$s = r + c * z$

s

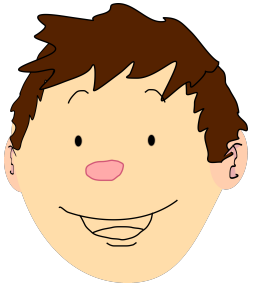
$g^s =^? \mathrm{A} * \mathrm{y}^c$

$\mathrm{A} * \mathrm{y}^c = g^r * g^{x*c}$

$g^r * g^{x*c} = g^{r+x*c}$

Non-Interactive Zero-Knowledge (NIZK)

I know x such that $g^x = y$



$r \leftarrow \mathbb{Z}_p$

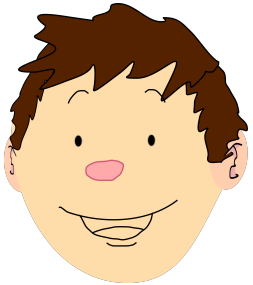
$A = g^r$
$c = H(A, y)$
$s = r + c * z$

$\pi = (A, c, s)$

$g^s =^? A * y^c$
$c =^? H(A, y)$

Schnorr signature

I know x such that $g^x = y$



$r \leftarrow \mathbb{Z}_p$

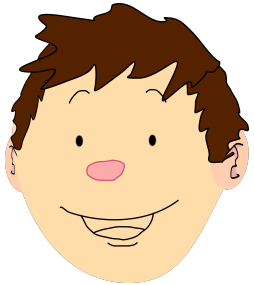
$A = g^r$
$c = H(A, y)$
$s = r + c * z$

$\pi = (A, c, s)$

$g^s =^? A * y^c$
$c =^? H(A, y)$

Schnorr signature

I know x such that $g^x = y$



$r \leftarrow \mathbb{Z}_p$

$A = g^r$
$c = H(A, y, M)$
$s = r + c * z$

$\sigma = (A, c, s), M$

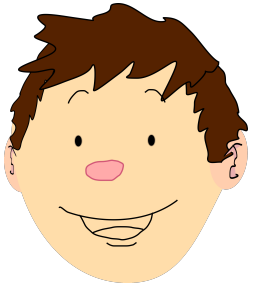
$g^s =^? A * y^c$
$c =^? H(A, y, M)$

Sigma protocols

- Good for NIZKs in public key systems
- Range proofs
- Proofs of solvency (Dagher et al. 15)
- Not good for more complicated statements

Proofs for complex statement

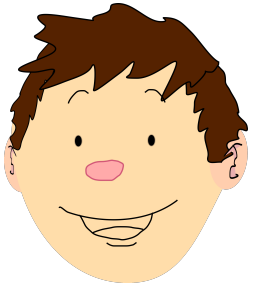
I know x such that H(x)=y/
This is the correct blockchain



π

Goal: Succinct proofs

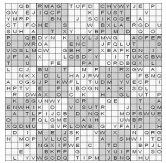
I know x such that H(x)=y/
This is the correct blockchain

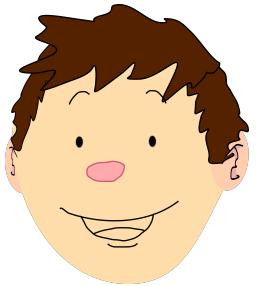


π

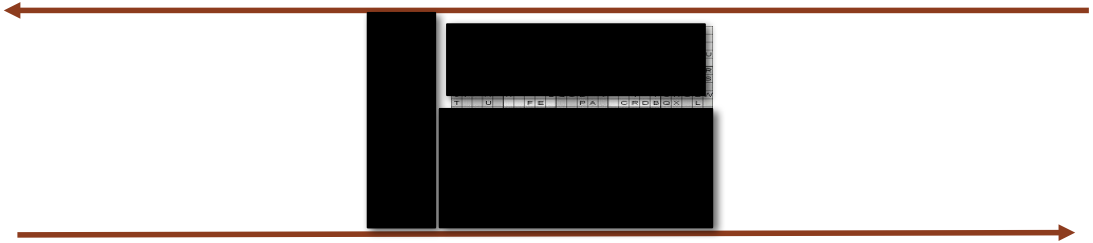
Proof is a 100 bytes no matter what the statement is
Verifying it takes ms

Preprocessing SNARK

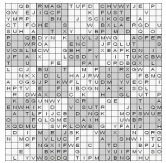




Open Row 7

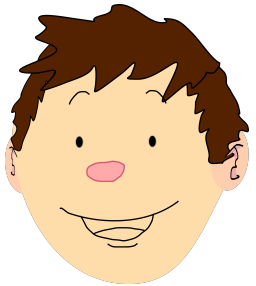


Preprocessing SNARK: Idea sent queries once



This can be reused

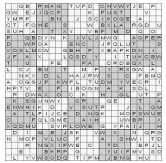
Open Row 7,Row 13, Column 4, Box 1 and the perm



Special compression function to compress answers (Pairing)

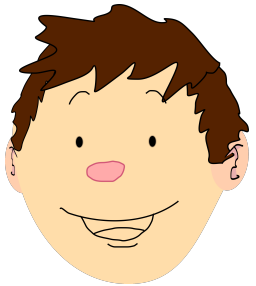
Compressed response: π

Preprocessing SNARK: Idea sent queries once



This can be reused

Open Row 7,Row 13, Column 4, Box 1 and the perm

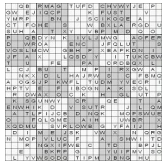


If Alice knows queries
She can cheat

Special compression function to compress answers (Pairing)

Compressed response: π

Preprocessing SNARK: Encrypt queries



This can be reused

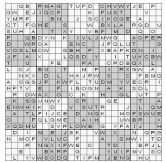
Open Row 7,Row 13, Column 4, Box 1 and the perm

If Alice knows queries
She can cheat

Special compression function to compress answers (Pairing)

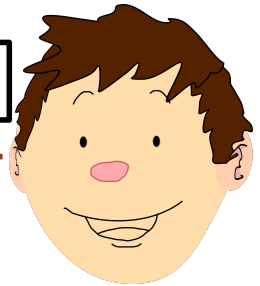
Compressed response: π

Preprocessing SNARK: Encrypt queries



This can be reused

Open Row 7,Row 13, Column 4, Box 1 and the perm



Special compression function to compress answers (Pairing)

Compressed response: π

Preprocessing SNARK: Trusted Setup

Special compression function to compress answers (Pairing)

Proving slow

Encrypted Queries

Setup slow

Short Encrypted Answers

Verify π Using encrypted answers

Verification fast

Compressed response: π

Preprocessing SNARK: Malicious Setup

Special compression function to compress answers (Pairing)

Encrypted Queries

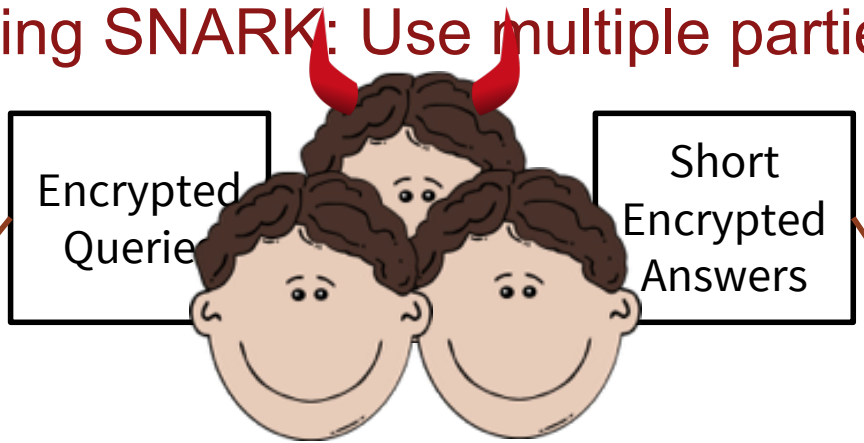
Short Encrypted Answers

Verify π Using encrypted answers

Can create cheating proofs

Compressed response: π

Preprocessing SNARK: Use multiple parties



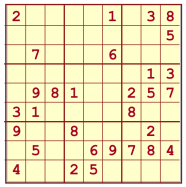
Special compression function to compress answers (Pairing)

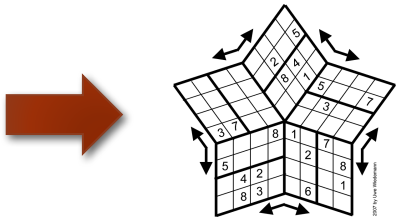
ZCash did this

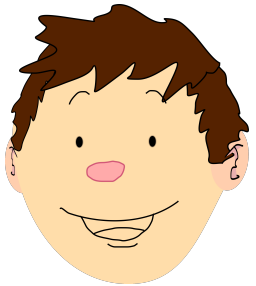
Verify π Using encrypted answers

Compressed response: π

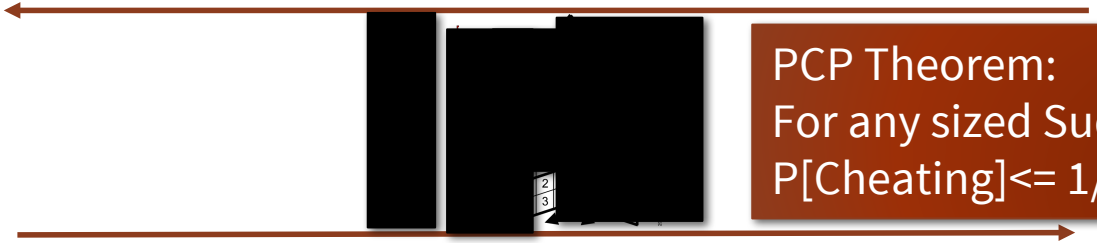
PCP Theorem





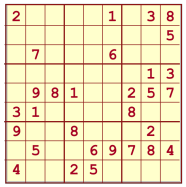


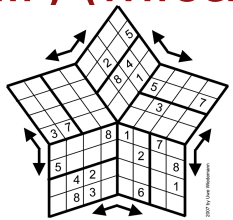
Open 2 fields



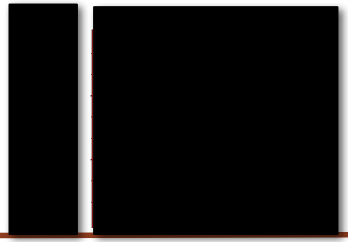
PCP Theorem:
For any sized Sudoku,
P[Cheating]<= 1/3

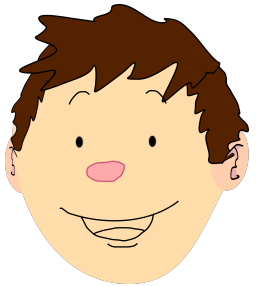
CS-Proofs (use Fiat-Shamir) (Micali 91)



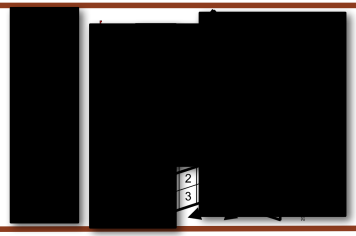


Commit:



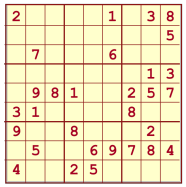


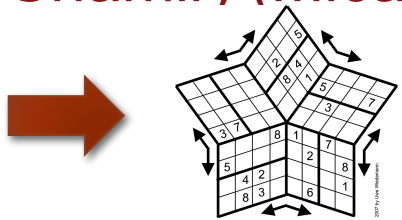
Open 2 fields=H(commit)



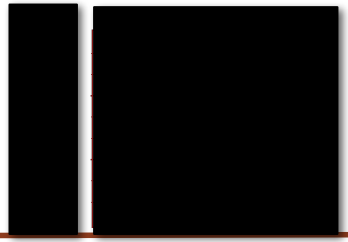
PCP Theorem:
For any sized Sudoku,
P[Cheating]<= 1/3

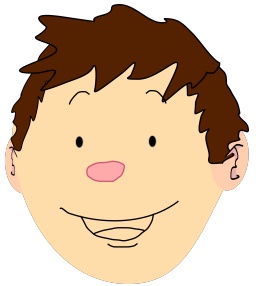
CS-Proofs (use Fiat-Shamir) (Micali 91)



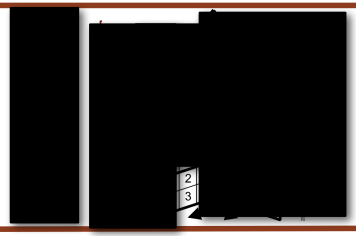


Commit:



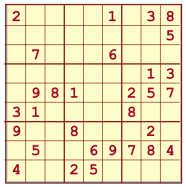


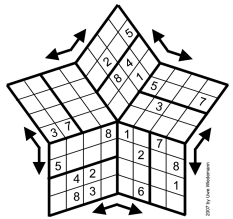
Open 2 fields=H(commit)



Not practical

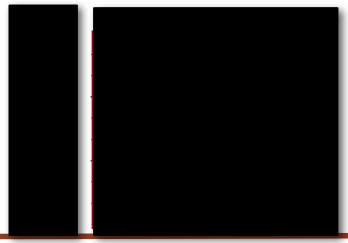
STARKs (Ben-Sasson 17)

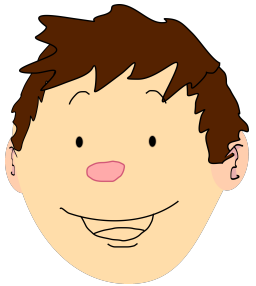




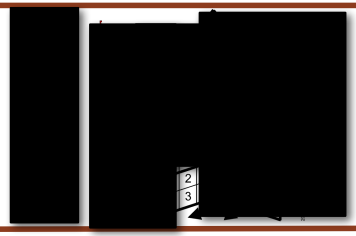
Making strides
2^{16} cycles
131 GB Ram usage
1.8 MB proofs

Commit:





Open 2 fields=H(commit)



Dreaming of STARKs

- ZCash without trusted setup
- Blockchain aggregation (doesn't need Zero-Knowledge)
- Confidential smart contracts
- Resolving verifiers dilema
- Generic verifiable computation (Outsourcing)
- Very active research area but still not there

Bulletproofs (Short proofs but linear verification)

- Bünz et al. 17 based on Bootle et al. 16
- Proofs are very short (log(n) for statement of size n)
- Verification is like proving (slow)
- Replacement for Sigma protocols
- No trusted setup!
- Just discrete log assumption

Bulletproofs (What is it good for?)

- Range proofs for confidential transactions/Mimblewimble
  - 670 bytes instead of 4 KB per range proof
  - Aggregation: Two range proofs 736 bytes vs. 8 KB
  - 16 range proofs 928 bytes vs. 61KB
  - Mimblewimble size: 17 GB vs 160 GB
- Built in: simple CoinJoin protocol for combining confidential transactions
- Solvency proofs
- Verifiable shuffles
- NIZKs for Smart Contracts
- …
- http://web.stanford.edu/~buenz/pubs/bulletproofs.pdf

References (Signatures)

- BLS https://www.iacr.org/archive/asiacrypt2001/22480516.pdf
- ECDSA Threshold: https://eprint.iacr.org/2016/013.pdf
- BLS/Schnorr Threshold: https://dl.acm.org/citation.cfm?id=359176
- Ring Signatures: https://www.iacr.org/archive/asiacrypt2001/22480516.pdf
- Blind Signatures: http://blog.koehntopp.de/uploads/Chaum.BlindSigForPayment.1982.PDF

References (Proofs)

- Provisions: https://eprint.iacr.org/2015/1008.pdf
- Zero-Knowledge contingent payments: http://stevengoldfeder.com/papers/ZKCSP.pdf
- Zero-cash: http://zerocash-project.org/media/pdf/zerocash-extended-20140518.pdf
- ZK-SUDOKU: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/sudoku_abs.html
- Sigma Protocols: ftp://ftp.inf.ethz.ch/pub/crypto/publications/CraDam98.pdf
- SNARKs:
  - http://www0.cs.ucl.ac.uk/staff/J.Groth/ShortNIZK.pdf (Groth first SNARK)
  - https://eprint.iacr.org/2012/215.pdf (GGPR SNARKs of today)
  - https://eprint.iacr.org/2013/279.pdf (Pinocchio the most used)
  - https://eprint.iacr.org/2013/507 (SNARKs for C)
- PCPs: http://people.eecs.berkeley.edu/~alexch/classes/CS294-S2017.html
- CS Proofs: https://www.computer.org/csdl/proceedings/focs/1994/6580/00/0365746.pdf
- STARKs: https://cyber.stanford.edu/sites/default/files/elibensasson.pdf
- Bulletproofs: http://web.stanford.edu/~buenz/pubs/bulletproofs.pdf

Thank you!

BUENZ@CS.STANFORD.EDU

http://web.stanford.edu/~buenz/pubs/bulletproofs.pdf