WI is Not Enough

Zero-Knowledge Contingent (Service) Payments Revisited

Georg Fuchsbauer

Inria INVENTEURS DU MONDE NUMÉRIQUE

ENS ÉCOLE NORMALE SUPÉRIEURE

CCS'19

London, 12 November 2019

Overview

Zero-knowledge contingent payments

fair exchange of goods for Bitcoin

- proposed by Maxwell 2011
- implemented by Bowe and Maxwell 2016

Campanelli, Gennaro, Goldfeder and Nizzardo (CCS'17)

- showed attack
- proposed efficient fixes

Overview

Zero-knowledge contingent payments

fair exchange of goods for Bitcoin

- proposed by Maxwell 2011
- implemented by Bowe and Maxwell 2016

Campanelli, Gennaro, Goldfeder and Nizzardo (CCS'17)

- showed attack
- proposed efficient fixes

This work

- show that efficient fixes are flawed

Fair exchange

Seller

Buyer

Fair exchange

Seller

Buyer

Fair exchange

Seller

Buyer

impossible without trusted party

Fair exchange of digital goods

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

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

Seller

Buyer: BTC

Fair exchange of digital goods

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

s

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

$V(\cdot) = ?$

Seller

s such that $V(s) = 1$

Buyer: BTC

Fair exchange of digital goods

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

s

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

$V(\cdot) = ?$

Seller

s such that $V(s) = 1$

Buyer: BTC

leverage trust in blockchain?

Smart contracts

Seller

Buyer: ETH

Ethereum:
Turing-compl. language

Smart contracts

Seller

Buyer: ETH

Ethereum:
Turing-compl. language

`contract`

pay whoever presents s such that $V(s) = 1$

Smart contracts

Seller

s

Buyer: ETH

Ethereum:
Turing-compl. language

`contract`

pay whoever presents s such that $V(s) = 1$

Smart contracts

Seller

s

pay

Buyer: ETH

Ethereum:
Turing-compl. language

`contract`

pay whoever presents s such that $V(s) = 1$

Smart contracts

Seller

s

pay

Buyer: ETH

Ethereum:
Turing-compl. language

`contract`

pay whoever presents s such that $V(s) = 1$

expensive for complex $V(\cdot)$

Smart contracts

Seller

Buyer: ETH

solution public

s

pay

`contract`

pay whoever presents s such that $V(s) = 1$

expensive for complex $V(\cdot)$

Ethereum:
Turing-compl. language

Bitcoin

Seller

Buyer: BTC

Bitcoin:
restricted scripting language
e.g. `Pay-to-PubkeyHash`

Bitcoin

Seller

Buyer: BTC

contract

pay whoever presents x such that $H(x) = y$

SHA-256

Bitcoin:
restricted scripting language
e.g. `Pay-to-PubkeyHash`

Zero-knowledge contingent payments [Maxwell'11]

Seller: s

Buyer: BTC

Zero-knowledge contingent payments

$$Enc_k(s),\ y = H(k)$$

Seller: s

Buyer: BTC

Zero-knowledge contingent payments

$Enc_k(s), y = H(k)$

Seller: s

Buyer: BTC

contract

pay whoever presents x such that $H(x) = y$

Zero-knowledge contingent payments

$Enc_k(s), y = H(k)$

k

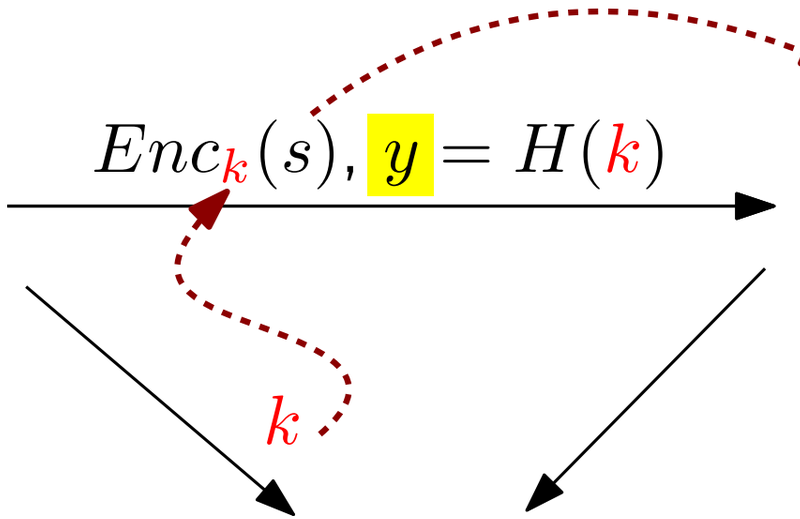
Seller: s

Buyer: BTC

contract

pay whoever presents x such that $H(x) = y$

Zero-knowledge contingent payments



Seller: s

Buyer: BTC

contract

pay whoever presents x such that $H(x) = y$

Zero-knowledge contingent payments

proof π

$Enc_k(s),\ y = H(k)$

k

Seller: s

Buyer: BTC

```
contract

pay whoever
presents x
such that
H(x) = y
```

Non-interactive proofs

$\rightarrow$ I know a **witness** w

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

for **statement** x

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

π

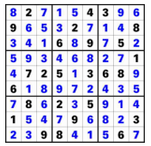
✓ / ✗

Prover: x, w

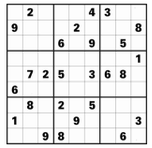
Verifier: x

Non-interactive proofs

$\rightarrow$ I know a **witness** w



for **statement** x

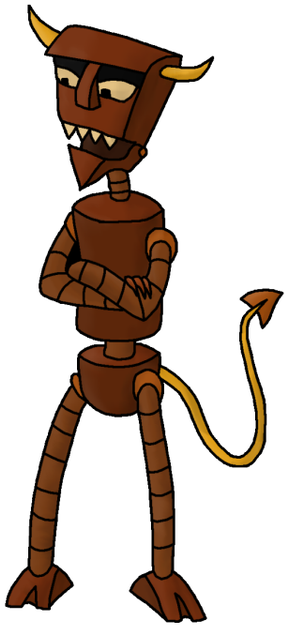


π

Zero knowledge:

nothing is revealed about w

[GMR'85, BFM'88]



Prover: x, w

Verifier: x

Non-interactive proofs

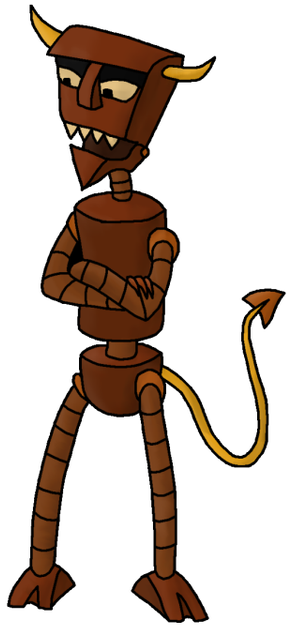
common reference string
generated by trusted party

π

Zero knowledge:

nothing is revealed about w

[GMR'85, BFM'88]



Prover: x, w

Verifier: x

Non-interactive proofs

common reference string
generated by trusted party

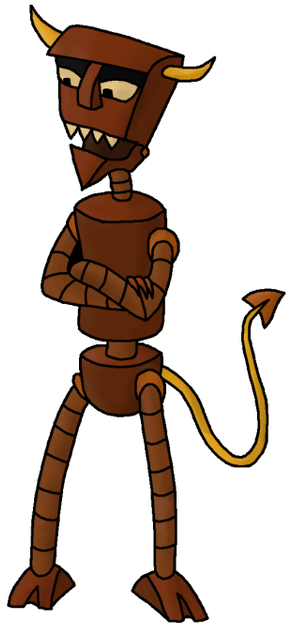
crs

π

Witness-indistinguishability:

$$\pi[w] \approx \pi[w']$$

[FS'90]



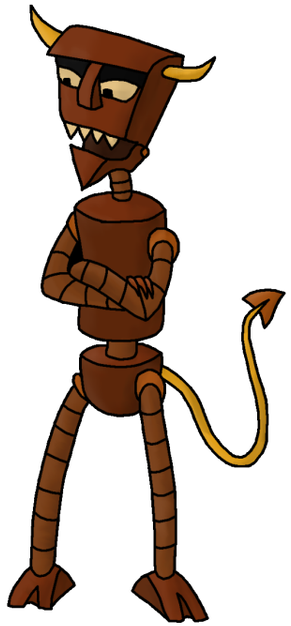
Prover: x, w

Verifier: x

Subversion-resistant proofs

crs

π



Subversion zero knowledge:

nothing is revealed about w

[BFS'16]

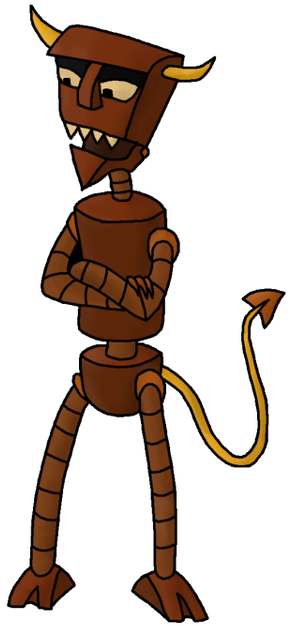
Prover: x, w

Verifier: x

Subversion-resistant proofs

crs

π



Subversion WI:

$$\pi[w] \approx \pi[w']$$

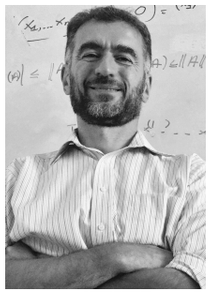
[BFS'16]

Prover: x, w

Verifier: x

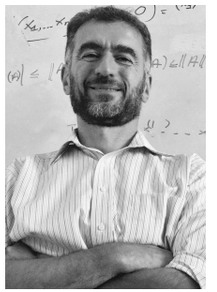
Zero-knowledge SNARKs

- **S**uccinct **N**on-interactive **AR**gument of **K**nowledge [GGPR'13]
- most efficient general NIZK proofs
- used in ZCASH [BCGGMTV'14]
  - fully **anonymous** cryptocurrency



Zero-knowledge SNARKs

- **S**uccinct **N**on-interactive **AR**gument of **K**nowledge [GGPR'13]
- most efficient general NIZK proofs



- used in ZCASH [BCGGMTV'14]
  - fully **anonymous** cryptocurrency

- zk-SNARKs can be made
  subversion-zero-knowlege [F'18]
  if prover checks well-formedness of CRS

Zero-knowledge contingent payments

Seller: s

$$Enc_k(s),\ H(k),\ \pi$$

zero-knowledge

SNARK

Buyer: BTC

Zero-knowledge contingent payments

Seller: s

$$Enc_k(s),\ H(k),\ \pi$$

zero-knowledge

SNARK

Buyer: BTC

Zero-knowledge contingent payments

Seller: s

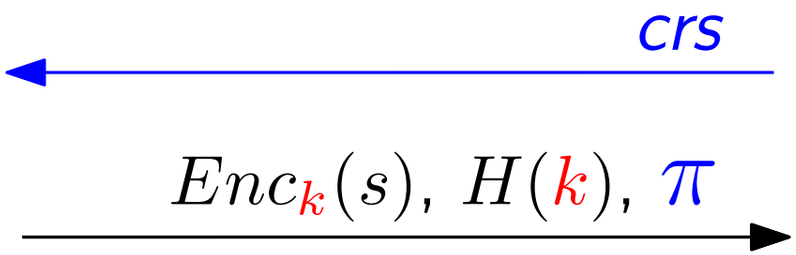
crs

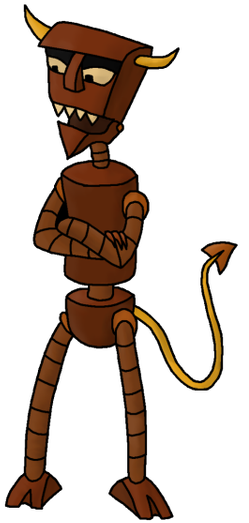
$Enc_k(s),\ H(k),\ \pi$

Buyer: BTC

Zero-knowledge contingent payments

Seller: s

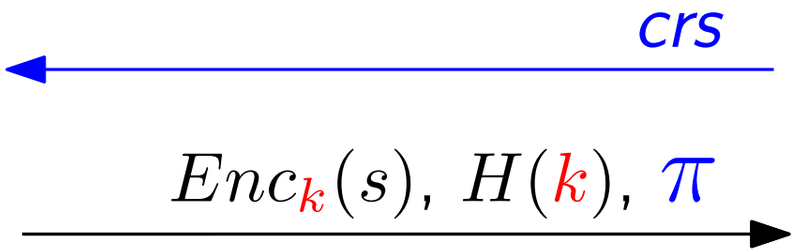


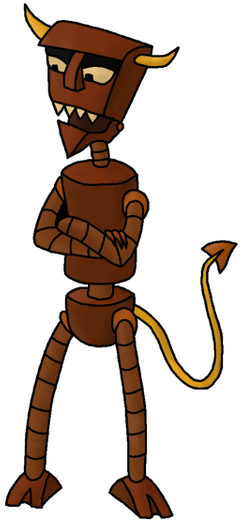


Subversion zero-knowledge?

Zero-knowledge contingent payments

Seller: s

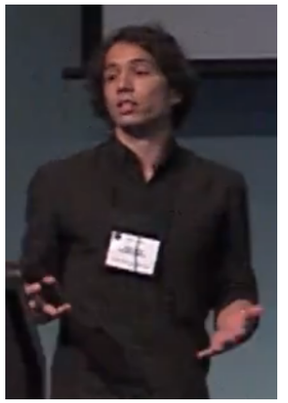




~~Subversion zero-knowledge?~~

- Campanelli, Gennaro, Goldfeder, Nizzardo (CCS'17) show **CRS-subversion attack**:

  $\Rightarrow$ obtain information on s



Proposed fixes

Fixes proposed by [CGGN'17]:

- use subversion-zk SNARKs [F'18]
- use MPC to compute CRS [BGG'18]

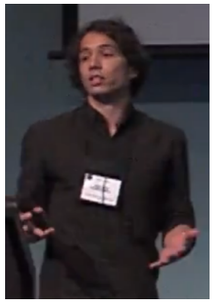
Proposed fixes

Fixes proposed by [CGGN'17]:

- use subversion-zk SNARKs [F'18] > 1 hour
- use MPC to compute CRS [BGG'18] > 3 hours

(original pay-to-sudoku: 1 minute)

Proposed fixes



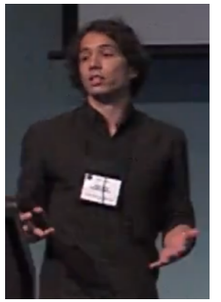
Fixes proposed by [CGGN'17]:

- use subversion-zk SNARKs [F'18] > 1 hour
- use MPC to compute CRS [BGG'18] > 3 hours

(original pay-to-sudoku: 1 minute)

- **"minimal checks"** to achieve subversion-WI
- **new protocol** from subversion-WI proofs (2 minutes)

Proposed fixes



Fixes proposed by [CGGN'17]:

- use subversion-zk SNARKs [F'18] > 1 hour
- use MPC to compute CRS [BGG'18] > 3 hours

(original pay-to-sudoku: 1 minute)

~~**"minimal checks"** to achieve subversion-WI~~

~~**new protocol** from subversion-WI proofs~~ (2 minutes)

"Minimal checks"

Subversion-zk SNARKs [F'18]

- **check** of **all** CRS elements using pairings (elliptic curves) > 1 hour

"Minimal checks"

Subversion-zk SNARKs [F'18]

- **check** of **all** CRS elements using pairings (elliptic curves) > 1 hour

[CGGN'17]

- "if **certain CRS elements non-zero** then WI even under malicious CRS"

"minimal checks"

"Minimal checks"

Subversion-zk SNARKs [F'18]

- **check** of **all** CRS elements using pairings (elliptic curves) > 1 hour

[CGGN'17]

- "if **certain CRS elements non-zero** then WI even under malicious CRS"

"minimal checks"

Our attack:

- change "i-th" CRS element $\Longrightarrow$ proofs valid iff $w_i = 0$

"Minimal checks"

Subversion-zk SNARKs [F'18]

- **check** of **all** CRS elements using pairings (elliptic curves) > 1 hour

[CGGN'17]

- "if **certain CRS elements non-zero** then WI even under malicious CRS"

"minimal checks"

Our attack:

- change "i-th" CRS element $\Longrightarrow$ proofs valid iff $w_i = 0$

$\rightarrow$ **breaks subversion-WI**

$\rightarrow$ consistency of **all elements** must be checked

Zero-knowledge contingent payments

Seller: s

crs

c, y, π

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

Buyer: BTC

Zero-knowledge contingent payments

Seller: s

crs

c, y, π

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

Buyer: BTC

what if buyer only wants to know if solution **exists**?

e.g. seller makes proof that it stores client's data

Zero-knowledge contingent payments

crs

c, y, π

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

Seller: s

Buyer: BTC

π **is** proof of existence/retrievability, . . .

what if buyer only wants to know if solution **exists**?

e.g. seller makes proof that it stores client's data

Zero-knowledge contingent **service** payments [CGGN'17]

Seller: s

crs

c, y, π

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

or $V(Dec_k(c)) = 0$ and $y = H'(k)$

Buyer: BTC

Zero-knowledge contingent **service** payments

Seller: s

crs

c, y, π

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

or $V(Dec_k(c)) = 0$ and $y = H'(k)$

as before

Buyer: BTC

Zero-knowledge contingent **service** payments

Seller: s

crs

c, y, π

Buyer: BTC

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

or $V(Dec_k(c)) = 0$ and $y = H'(k)$

as before

no solution, **no payment!**

Zero-knowledge contingent **service** payments

Seller: s

crs

c, y, π

Buyer: BTC

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

or $V(Dec_k(c)) = 0$ and $y = H'(k)$

as before

no solution, **no payment!**

clawfreeness:
hard to find k_0, k_1 :
$H'(k_0) = H(k_1)$

Zero-knowledge contingent **service** payments

Seller: s

crs

c, y, π

Buyer: BTC

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

or $V(Dec_k(c)) = 0$ and $y = H'(k)$

Claim [CGGN'17]: π only needs to be WI

clawfreeness:
hard to find k_0, k_1 :
$H'(k_0) = H(k_1)$

Zero-knowledge contingent **service** payments

Seller: s

crs

c, y, π

Buyer: BTC

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

or $V(Dec_k(c)) = 0$ and $y = H'(k)$

~~**Claim:**~~

Attack: WI is not enough

clawfreeness:
hard to find k_0, k_1 :
$H'(k_0) = H(k_1)$

Zero-knowledge contingent **service** payments

Seller: s

Buyer: BTC

crs ←

c, y, π →

π: I know k s.t.

$V(Dec_k(c)) = 1$ and $y = H(k)$

or $V(Dec_k(c)) = 0$ and $y = H'(k)$

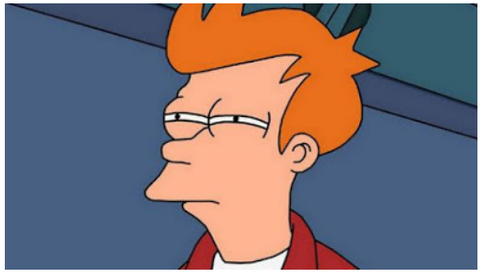
~~**Claim:**~~ **Attack:** WI is not enough

Proof: Prove′:

- $s := \mathrm{Dec}_k(c)$
- if $V(s) = 1$ then return $\pi \| s$
- return $\pi \| 0$ □

clawfreeness:
hard to find k_0, k_1 :
$H'(k_0) = H(k_1)$

Conclusion



- zk contingent payments and
- zk contingent service payments

  require *subversion-ZK* proofs

  (subversion) WI is not enough

- costly CRS checks necessary

  even for subversion WI

  "minimal checks" are not enough

THANK YOU!

QUESTIONS?