Auditable Decentralized Confidential Payment System

Yu Chen
Shandong University

based on works on ESORICS 2020 and FinCrypto 2020 competition
http://eprint.iacr.org/2019/319
https://github.com/yuchen1024/libPGC
https://github.com/yuchen1024/Kunlun/adct

3rd CSDP 2021.08.19

Outline

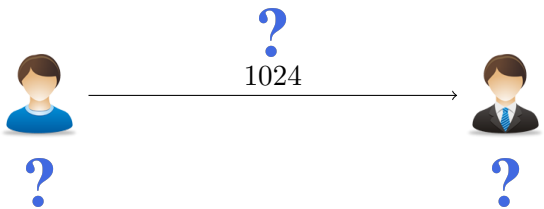
1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

Outline

1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

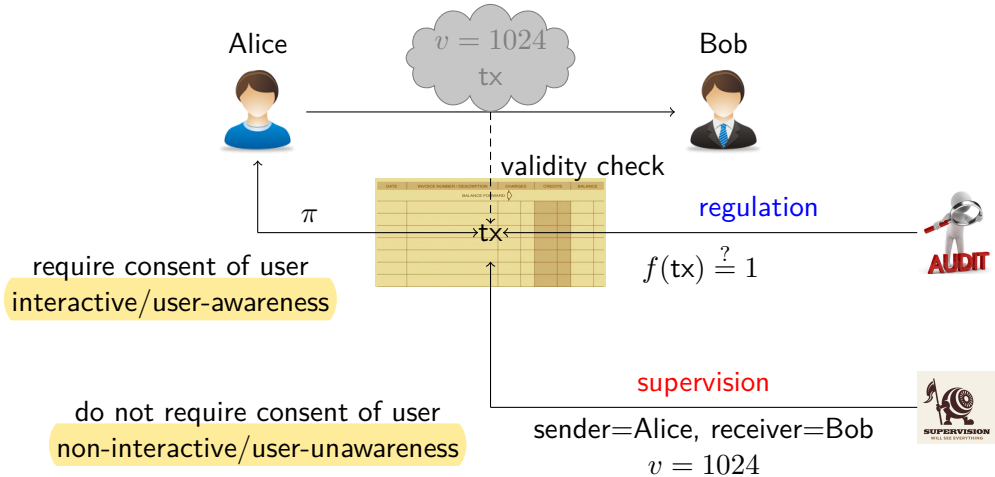
Privacy in Payment System

external observer cannot learn the transfer amount

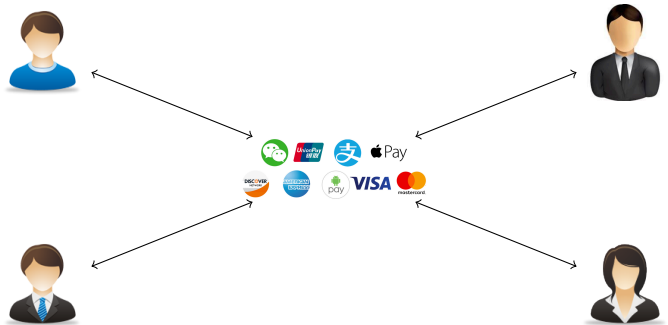


external observer cannot identify the sender and recipient

Auditing in Payment System



Centralized Payment System



- txs are kept on a private ledger only known to the center
- the center is in charge of validity check as well as protecting privacy and conducting audit

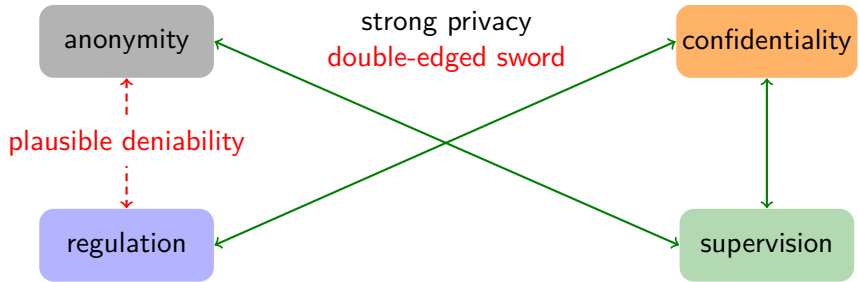
Decentralized Payment System (Blockchain-based Cryptocurrencies)

- txs are kept on a global distributed public ledger — the blockchain
- to ensure public verifiability, Bitcoin and Ethereum simply expose all tx information in public $\rightsquigarrow$ no privacy

Motivation

Privacy and Auditability are crucial in any financial system, we want to know:

In the decentralized setting, can we have the good of both?



In this work, we trade anonymity for auditing, propose the first

decentralized confidential payment (DCP) system (in the account-based model)
support regulation and supervision

Outline

1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

Desired Functionality and Security

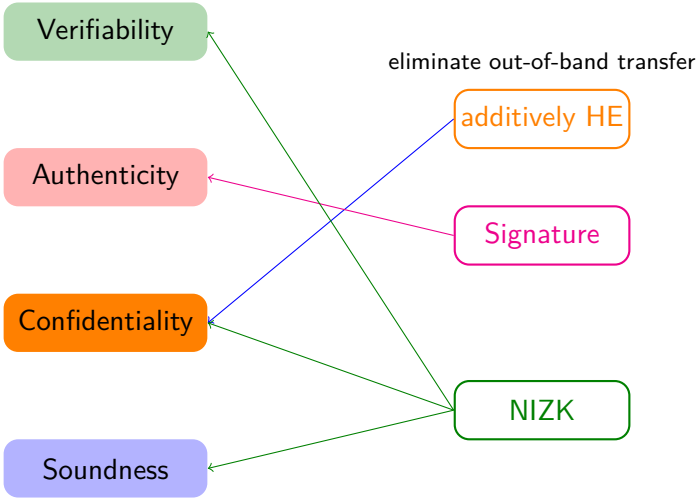
Verifiability	validity of txs are publicly verifiable
Authenticity	only owner can generate tx; nobody else can forge
Confidentiality	external observer learns the transfer amount
Soundness	nobody cannot generate an illegal tx that passes validity check
Auditability	user cannot cheat and regulation does not leak more info other than auditing result
Supervision	auditor can see everything, but unable to breach authenticity

Formal security model for ADCP is quite challenging

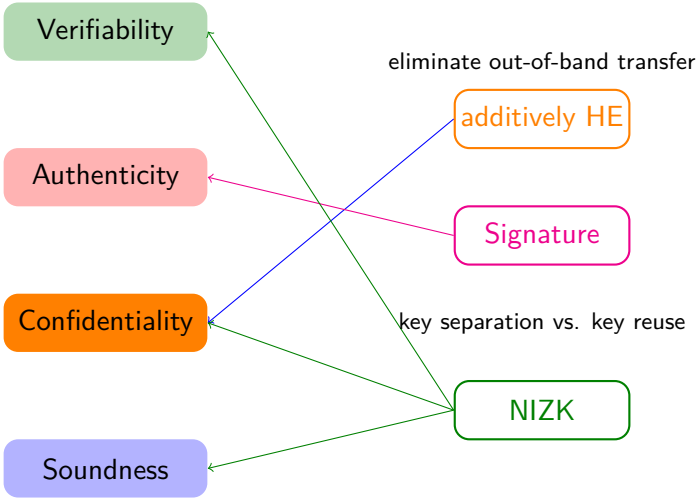
- powerful enough to capture all possible real-world attacks
- clean and handy to use

We refer to the backup slides for the details.

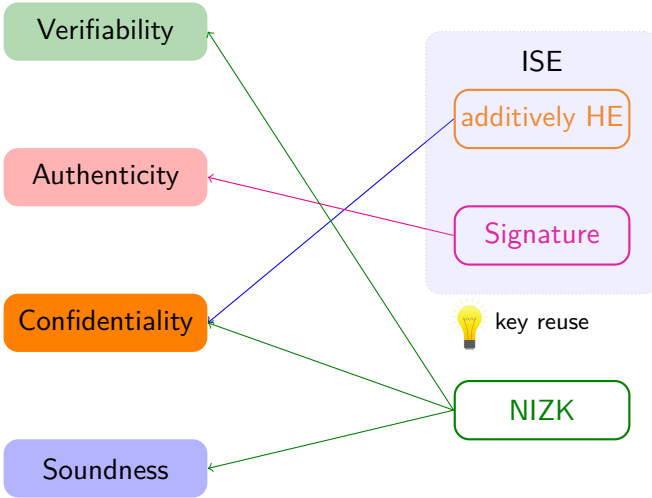
Choice of Building Blocks



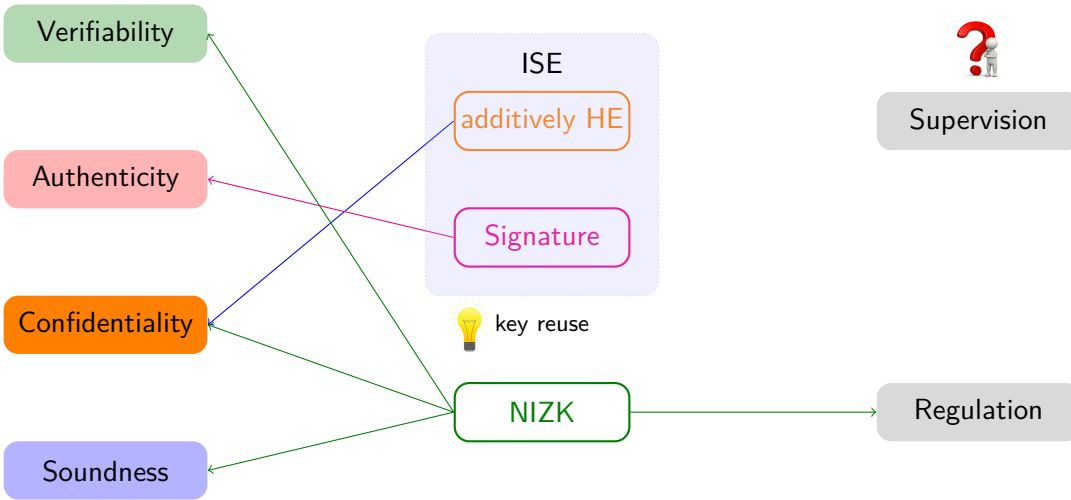
Choice of Building Blocks



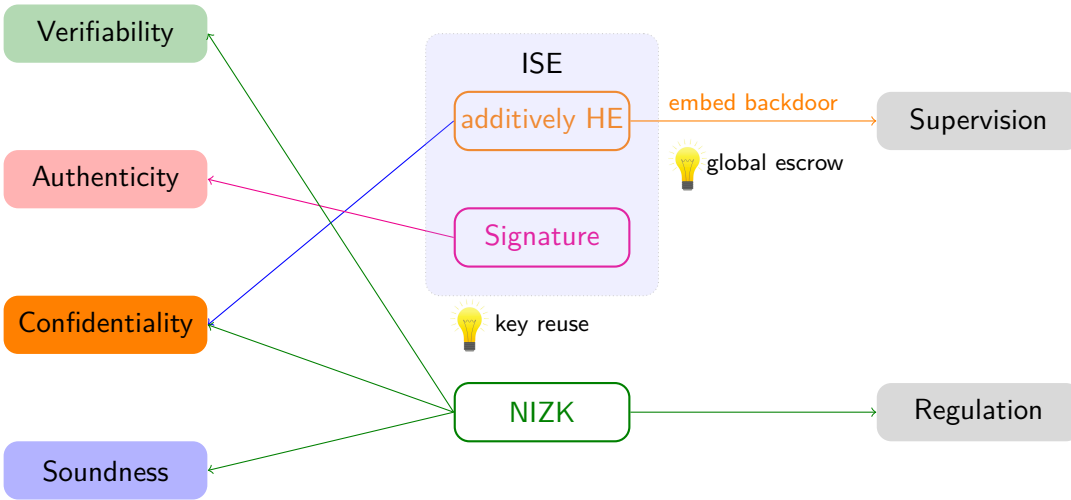
Choice of Building Blocks



Choice of Building Blocks



Choice of Building Blocks



Generic Construction of ADCP

$$\mathsf{Setup}(1^\lambda) \to (pp, sp)$$

$\mathsf{ISE.Setup}(1^\lambda) \to pp_{\mathsf{ise}}$, $\mathsf{NIZK.Setup}(1^\lambda) \to pp_{\mathsf{nizk}}$
$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_a, sk_a)$

embed backdoor for supervision

Generic Construction of ADCP

$\mathsf{Setup}(1^\lambda) \to (pp, sp)$

$\mathsf{ISE.Setup}(1^\lambda) \to pp_{\mathsf{ise}}$, $\mathsf{NIZK.Setup}(1^\lambda) \to pp_{\mathsf{nizk}}$
$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_a, sk_a)$

$\mathsf{CreateAcct}(v_s, \mathsf{sn}_s)$

$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_s, sk_s)$
$\mathsf{ISE.Enc}(pk_s, v_s) \to \tilde{C}_s$

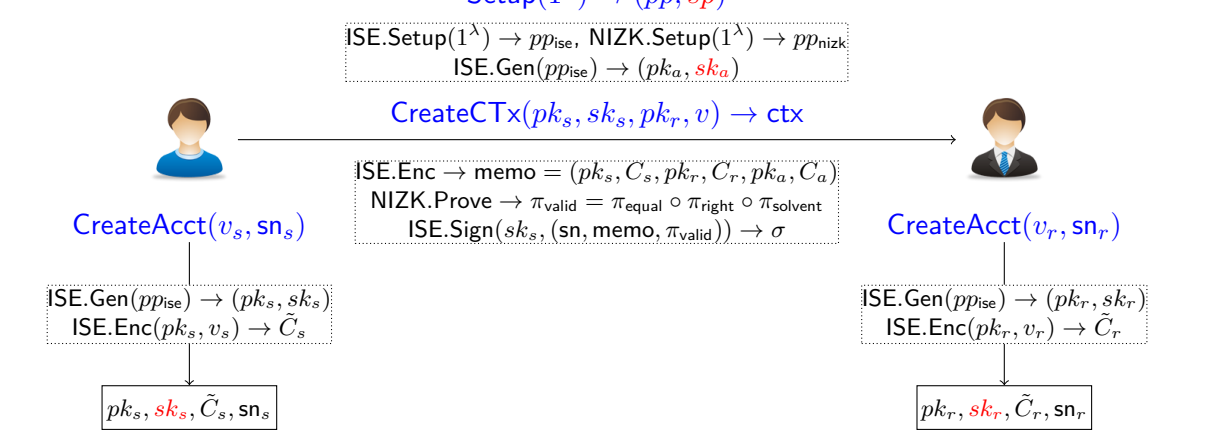
$pk_s, sk_s, \tilde{C}_s, \mathsf{sn}_s$

$\mathsf{CreateAcct}(v_r, \mathsf{sn}_r)$

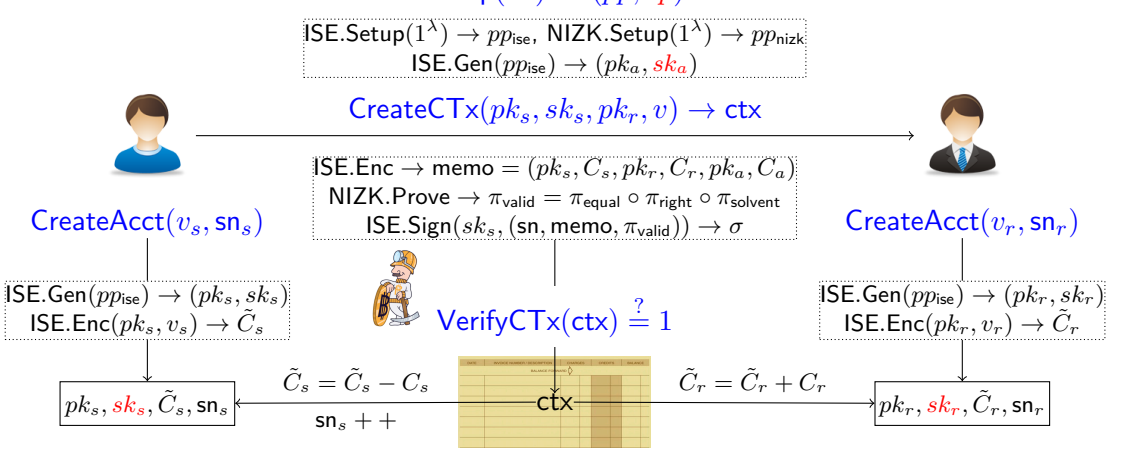
$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_r, sk_r)$
$\mathsf{ISE.Enc}(pk_r, v_r) \to \tilde{C}_r$

$pk_r, sk_r, \tilde{C}_r, \mathsf{sn}_r$

Generic Construction of ADCP



Generic Construction of ADCP



Generic Construction of ADCP

$\mathsf{Setup}(1^\lambda) \to (pp, sp)$

$\mathsf{ISE.Setup}(1^\lambda) \to pp_{\mathsf{ise}}$, $\mathsf{NIZK.Setup}(1^\lambda) \to pp_{\mathsf{nizk}}$
$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_a, sk_a)$

$\mathsf{CreateCTx}(pk_s, sk_s, pk_r, v) \to \mathsf{ctx}$

$\mathsf{ISE.Enc} \to \mathsf{memo} = (pk_s, C_s, pk_r, C_r, pk_a, C_a)$
$\mathsf{NIZK.Prove} \to \pi_{\mathsf{valid}} = \pi_{\mathsf{equal}} \circ \pi_{\mathsf{right}} \circ \pi_{\mathsf{solvent}}$
$\mathsf{ISE.Sign}(sk_s, (\mathsf{sn}, \mathsf{memo}, \pi_{\mathsf{valid}})) \to \sigma$

$\mathsf{CreateAcct}(v_s, \mathsf{sn}_s)$

$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_s, sk_s)$
$\mathsf{ISE.Enc}(pk_s, v_s) \to \tilde{C}_s$

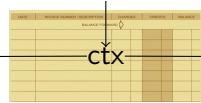
$\mathsf{VerifyCTx}(\mathsf{ctx}) \stackrel{?}{=} 1$

$\mathsf{CreateAcct}(v_r, \mathsf{sn}_r)$

$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_r, sk_r)$
$\mathsf{ISE.Enc}(pk_r, v_r) \to \tilde{C}_r$

$pk_s, sk_s, \tilde{C}_s, \mathsf{sn}_s$

$\tilde{C}_s = \tilde{C}_s - C_s$
$\mathsf{sn}_s + +$



$\tilde{C}_r = \tilde{C}_r + C_r$

$pk_r, sk_r, \tilde{C}_r, \mathsf{sn}_r$

$\mathsf{AuditCTx}(\pi_f, \{\mathsf{ctx}_i\}, f)$

$\mathsf{JustifyCTx}(sk, \{\mathsf{ctx}_i\}, f) \to \pi_f$

Generic Construction of ADCP

$\mathsf{Setup}(1^\lambda) \to (pp, sp)$

$\mathsf{ISE.Setup}(1^\lambda) \to pp_{\mathsf{ise}}$, $\mathsf{NIZK.Setup}(1^\lambda) \to pp_{\mathsf{nizk}}$
$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_a, sk_a)$

$\mathsf{CreateCTx}(pk_s, sk_s, pk_r, v) \to \mathsf{ctx}$

$\mathsf{ISE.Enc} \to \mathsf{memo} = (pk_s, C_s, pk_r, C_r, pk_a, C_a)$
$\mathsf{NIZK.Prove} \to \pi_{\mathsf{valid}} = \pi_{\mathsf{equal}} \circ \pi_{\mathsf{right}} \circ \pi_{\mathsf{solvent}}$
$\mathsf{ISE.Sign}(sk_s, (\mathsf{sn}, \mathsf{memo}, \pi_{\mathsf{valid}})) \to \sigma$

$\mathsf{CreateAcct}(v_s, \mathsf{sn}_s)$

$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_s, sk_s)$
$\mathsf{ISE.Enc}(pk_s, v_s) \to \tilde{C}_s$

$\mathsf{VerifyCTx}(\mathsf{ctx}) \stackrel{?}{=} 1$

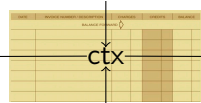
$\mathsf{CreateAcct}(v_r, \mathsf{sn}_r)$

$\mathsf{ISE.Gen}(pp_{\mathsf{ise}}) \to (pk_r, sk_r)$
$\mathsf{ISE.Enc}(pk_r, v_r) \to \tilde{C}_r$

$pk_s, sk_s, \tilde{C}_s, \mathsf{sn}_s$

$\tilde{C}_s = \tilde{C}_s - C_s$

$\mathsf{sn}_s + +$



$\tilde{C}_r = \tilde{C}_r + C_r$

$pk_r, sk_r, \tilde{C}_r, \mathsf{sn}_r$

$\mathsf{AuditCTx}(\pi_f, \{\mathsf{ctx}_i\}, f)$

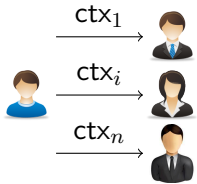
$\mathsf{JustifyCTx}(sk, \{\mathsf{ctx}_i\}, f) \to \pi_f$

$\mathsf{OpenCTx}(sp, \mathsf{ctx}) \to v$

Regulation

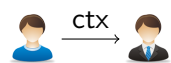
expressiveness of NIZK in use $\rightsquigarrow$ supported regulation policies

$f_{\mathsf{limit}} : \sum_{i=1}^{n} v_i < \ell$
anti-money laundering



$f_{\mathsf{rate}} : v_1/v_2 = \rho$
pay tax

$f_{\mathsf{open}} : v = v^*$
selective opening



Supervision

Supervision

STOC 1990

Naor-Yung
double enc paradigm

PKE
CPA $\rightsquigarrow$ CCA

Supervision

data structure of ctx

signed part

core memo info

π_{valid}

sk_a

sn	$pk_s, C_s, pk_r, C_r, pk_a, C_a$	π_{equal}	π_{right}	π_{solvent}	σ

aux info

triple enc paradigm $\Longrightarrow$ auditor's view = recipient's view

ensure supervision correctness

STOC 1990

extend

Naor-Yung
double enc paradigm $\longrightarrow$ PKE
CPA $\rightsquigarrow$ CCA

Outline

1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

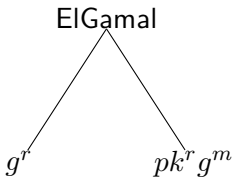
Disciplines in Mind

While ADCP framework is intuitive, secure and efficient instantiation requires clever choice and design of building blocks.

---

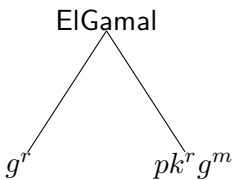
efficient		efficient ctx generation/verification compact ctx size
transparent setup		system does not require a trusted setup design case-tailored NIZK
simple & modular		build the system from reusable gadgets can be reused in other places

the initial attempt



Encryption Component of ISE

the initial attempt

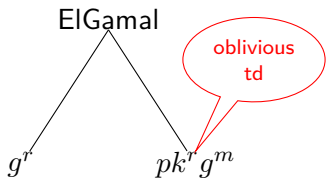


state-of-the-art

Bulletproofs

Encryption Component of ISE

the initial attempt

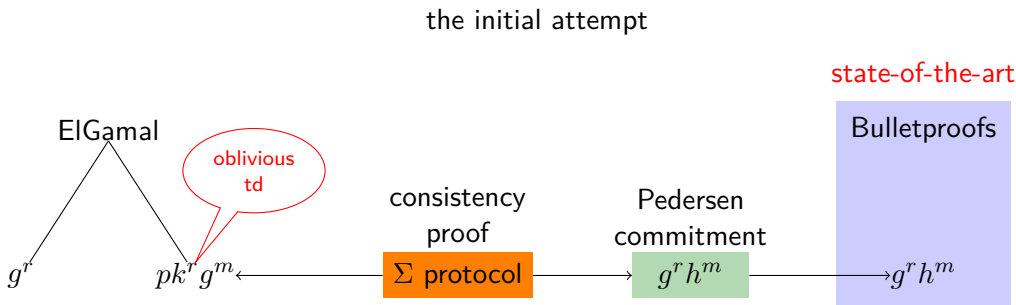


state-of-the-art

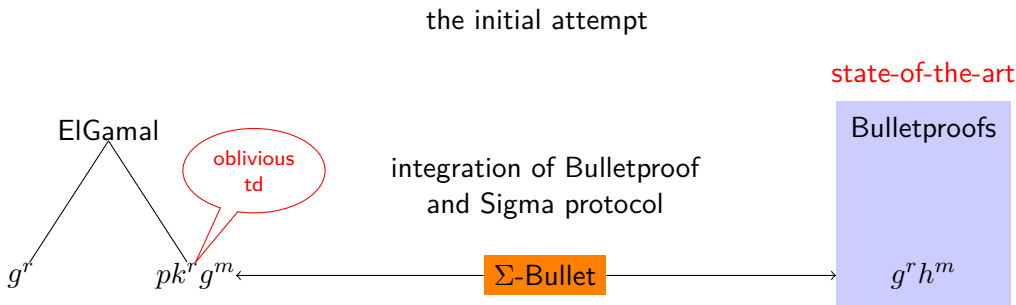
Bulletproofs

$g^r h^m$

Encryption Component of ISE

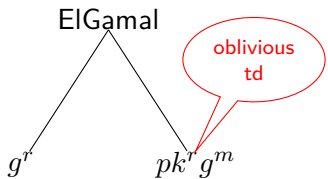


Encryption Component of ISE



Encryption Component of ISE

the initial attempt



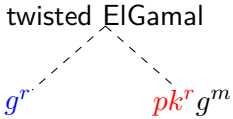
state-of-the-art

Bulletproofs

$g^r h^m$

simple and efficient, but not friendly to the state-of-the-art range proofs

Encryption Component of ISE: Twisted ElGamal



Encryption Component of ISE: Twisted ElGamal

Encryption Component of ISE: Twisted ElGamal

Encryption Component of ISE: Twisted ElGamal

- encode message over another generator h
- switch key encapsulation and session key
- advantages
  1. as secure and efficient as standard ElGamal;
  2. Bulletproofs-friendly: especially in the aggregated mode
  3. also friendly to other range proofs [CCS08, CKLR21] that accept Pedersen commitment as instance

Comparison to ElGamal

	size				efficiency		
ElGamal	pp	pk	sk	C	KeyGen	Enc	Dec
standard	$\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{Z}_p\|$	$\|2\mathbb{G}\|$	1Exp	3Exp+2Add	1Exp+1Add+1DLOG
twisted	$2\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$\|\mathbb{Z}_p\|$	$\|2\mathbb{G}\|$	1Exp	3Exp+2Add	1Exp+1Add+1DLOG

Related works [FMMO19, BAZB20] use brute-force algorithm to decrypt, we use Shanks's algorithm to speed decryption $\Rightarrow$ admits flexible time/space trade-off and parallelization!

Table: Costs of working with Bulletproofs between standard ElGamal and twisted ElGamal: an additional Pedersen commitment and a Sigma protocol for consistency.

ElGamal	size	efficiency
standard	$2\|\mathbb{G}\| + \|\mathbb{Z}_p\|$	4Exp+1Add
twisted	0	0

the saving could be tremendous when processing millions of data

Comparison to Paillier

Table: Twisted ElGamal vs. Paillier PKE (32-bit message space and 128-bit security)

timing (ms)	Setup	KeyGen	Enc	Dec	ReRand	Add	Sub	Scalar
Paillier	—	1644.53	32.211	31.367	—	0.0128	—	—
t-ElGamal	53s+5s	0.009	0.094	0.604	0.105	0.004	0.004	0.079

with 64MB lookup table to accelerate decryption $4 \sim 300\times$ speed up in computation efficiency

size (bytes)	public parameters	public key	secret key	ciphertext
Paillier	—	384	384	768
t-ElGamal	66	33	32	66

$10\times$ speed up in communication cost

Signature Component of ISE

We choose Schnorr signature as the signature component.

1. Setup and KeyGen of Schnorr signature are identical to those of twisted ElGamal.

   key reuse strategy ✓

2. Sign of Schnorr signature is irrelevant to Decrypt of twisted ElGamal:
   - Sign(sk, m): pick $r \xleftarrow{R} \mathbb{Z}_p$, set $A = g^r$, compute $e = \mathsf{H}(m, A)$, $z = r + sk \cdot e \bmod p$, output $\sigma = (A, z)$.

recall Schnorr signature is provably secure by modeling H as RO: simulating signature oracle by programing H without using sk $\Rightarrow$ signatures reveals zero-knowledge of sk

joint security ✓

We can also use ECDSA/SM2 signature schemes.

NIZK for L_{equal}

According to our ADCP framework and twisted ElGamal, L_{equal} can be written as:

$$\{(pk_i, X_i, Y_i)_{i\in[3]} \mid \exists r_1, r_2, r_3, v \text{ s.t. } X_i = pk_i^{r_i} \wedge Y_i = g^{r_i} h^v \text{ for } i = 1, 2, 3\}.$$

On statement $(pk_i, X_i, Y_i)_{i\in[3]}$, P and V interact as below:

1. P picks $a, b \xleftarrow{R} \mathbb{Z}_p$, sends $A_i = pk_i^{a_i}$, $B = g^a h^b$ to V.
2. V picks $e \xleftarrow{R} \mathbb{Z}_p$ and sends it to P as the challenge.
3. P computes $z_i = a + er_i$ for $i \in [3]$ and $t = b + ev$ using $w = (r_1, r_2, r_3, v)$, then sends (z_1, z_2, z_3, t) to V. V accepts iff the following four equations hold simultaneously:

$$pk_i^{z_i} = A_i X_i^e \quad (1)$$
$$g^{z_1} h^t = B Y_1^e \quad (2)$$

NIZK for L_{right}

According to our ADCP framework and twisted ElGamal, L_{right} can be written as:

$$\{(pk, X, Y) \mid \exists r, v \text{ s.t. } X = pk^r \wedge Y = g^r h^v \wedge v \in \mathcal{V}\}.$$

For ease of analysis, we additionally define L_{enc} and L_{range} as below:

$$L_{\mathsf{enc}} = \{(pk, X, Y) \mid \exists r, v \text{ s.t. } X = pk^r \wedge Y = g^r h^v\}$$
$$L_{\mathsf{range}} = \{Y \mid \exists r, v \text{ s.t. } Y = g^r h^v \wedge v \in \mathcal{V}\}$$

It is straightforward to verify that $L_{\mathsf{right}} \subset L_{\mathsf{enc}} \wedge L_{\mathsf{range}}$.

- Σ_{enc}: Sigma protocol for L_{enc}
- $\Lambda_{\mathsf{bullet}}$: Bulletproofs for L_{range}

DL relation between (g, h) is hard $\Rightarrow \Sigma_{\mathsf{enc}} \circ \Lambda_{\mathsf{bullet}}$ is SHVZK PoK for L_{right}

NIZK for L_{solvent}

According to our ADCP framework, L_{solvent} can be written as:

$$\{(pk, \tilde{C}, C) \mid \exists sk \text{ s.t. } (pk, sk) \in \mathsf{R}_{\mathsf{key}} \wedge \mathsf{ISE.Dec}(sk, \tilde{C} - C) \in \mathcal{V}\}.$$

$\tilde{C} = (\tilde{X} = pk^{\tilde{r}}, \tilde{Y} = g^{\tilde{r}} h^{\tilde{m}})$ encrypts $\tilde{m}$ of pk under $\tilde{r}$, $C = (X = pk^{r}, Y = g^{r} h^{v})$ encrypts v under r. Let $C' = (X' = pk^{r'}, Y' = g^{r'} h^{m'}) = \tilde{C} - C$, L_{solvent} can be rewritten as:

$$\{(pk, C') \mid \exists r', m' \text{ s.t. } C' = \mathsf{ISE.Enc}(pk, m'; r') \wedge m' \in \mathcal{V}\}.$$

Prove it as L_{right}? No! r' is unknown.

Solution: refresh-then-prove

1. refresh C' to C^* under fresh randomness $r^* \Leftarrow$ can be done with sk
2. prove $(C', C^*) \in L_{\mathsf{equal}} \Leftarrow$ Sigma protocol Σ_{ddh} (do not need r')
3. prove $C^* \in L_{\mathsf{right}}$

Bonus: two useful proof gadgets

twisted ElGamal + Bulletproofs: prove an encrypted message lies in specific range

- extremely useful in privacy-preserving applications: confidential transaction and secure machine learning

Bonus: two useful proof gadgets

twisted ElGamal + Bulletproofs: prove an encrypted message lies in specific range

- extremely useful in privacy-preserving applications: confidential transaction and secure machine learning

---

$$pk^r \qquad g^r h^m$$

prover is the sender of C
knows both r and m

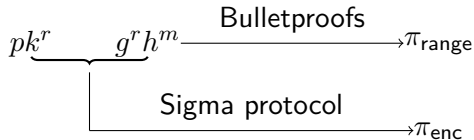
Bonus: two useful proof gadgets

twisted ElGamal + Bulletproofs: prove an encrypted message lies in specific range

- extremely useful in privacy-preserving applications: confidential transaction and secure machine learning

---

prover is the sender of C
knows both r and m



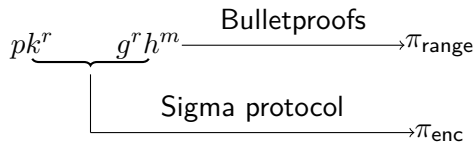
Bonus: two useful proof gadgets

twisted ElGamal + Bulletproofs: prove an encrypted message lies in specific range

- extremely useful in privacy-preserving applications: confidential transaction and secure machine learning

---

prover is the sender of C
knows both r and m



---

prover is the receiver of C
knows sk and thus m

$pk^{\boxed{r}}$ $g^{\boxed{r}} h^m$

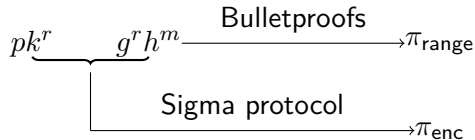
Bonus: two useful proof gadgets

twisted ElGamal + Bulletproofs: prove an encrypted message lies in specific range

- extremely useful in privacy-preserving applications: confidential transaction and secure machine learning

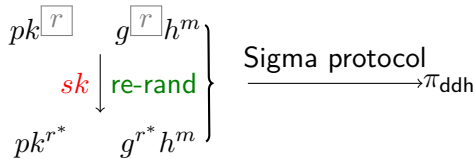
---

prover is the sender of C
knows both r and m



---

prover is the receiver of C
knows sk and thus m



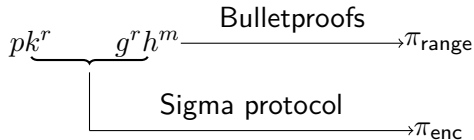
Bonus: two useful proof gadgets

twisted ElGamal + Bulletproofs: prove an encrypted message lies in specific range

- extremely useful in privacy-preserving applications: confidential transaction and secure machine learning

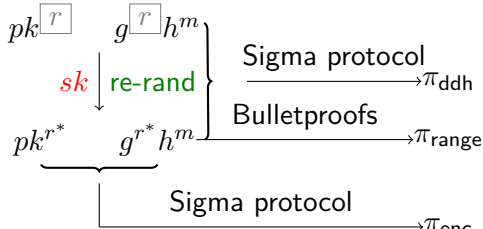
---

prover is the sender of C
knows both r and m



---

prover is the receiver of C
knows sk and thus m



NIZK for Auditing Policies: (1/2)

$$L_{\mathsf{limit}} = \{(pk, \{C_i\}_{1 \le i \le n}, a_{\max}) \mid \exists sk \text{ s.t.}$$
$$(pk, sk) \in \mathsf{R}_{\mathsf{key}} \wedge v_i = \mathsf{ISE.Dec}(sk, C_i) \wedge \sum_{i=1}^{n} v_i \le a_{\max}\}$$

P computes $C = \sum_{i=1}^{n} C_i$, proves $(pk, C) \in L_{\mathsf{solvent}}$ using Gadget-2

$$L_{\mathsf{open}} = \{(pk, C = (X, Y), v) \mid \exists sk \text{ s.t. } X = (Y/h^v)^{sk} \wedge pk = g^{sk}\}$$

$(pk, X, Y, v) \in L_{\mathsf{open}}$ is equivalent to $(Y/h^v, X, g, pk) \in L_{\mathsf{ddh}}$.

NIZK for Auditing Policies: (2/2)

$$L_{\mathsf{rate}} = \{(pk, C_1, C_2, \rho) \mid \exists sk \text{ s.t.}$$
$$(pk, sk) \in \mathsf{R}_{\mathsf{key}} \wedge v_i = \mathsf{ISE.Dec}(sk, C_i) \wedge v_1/v_2 = \rho\}$$

We assume $\rho = \alpha/\beta$, where α, β are positive integer much smaller than p.

Let $C_1 = (pk^{r_1}, g^{r_1} h^{v_1})$, $C_2 = (pk^{r_2}, g^{r_2} h^{v_2})$. P computes

$$C_1' = \beta \cdot C_1 = (X_1' = pk^{\beta r_1}, Y_1' = g^{\beta r_1} h^{\beta v_1})$$
$$C_2' = \alpha \cdot C_2 = (X_2' = pk^{\alpha r_2}, Y_2' = g^{\alpha r_2} h^{\alpha v_2})$$

Note $v_1/v_2 = \rho = \alpha/\beta$ iff $h^{\beta v_1} = h^{\alpha v_2}$. $(pk, C_1, C_2, \rho) \in L_{\mathsf{rate}}$ is equivalent to $(Y_1'/Y_2', X_1'/X_2', g, pk) \in L_{\mathsf{ddh}}$.

Thanks to nice algebra structure of twisted ElGamal, PGC supports efficient auditing for any policy that can be expressed as linear constraint over transfer amount and balance

Optimizations

sn	$pk_s, C_s, pk_r, C_r, pk_a, C_a$	$\pi_{\mathsf{equal}} \circ (\pi^1_{\mathsf{enc}} \circ \pi^1_{\mathsf{bullet}}) \circ (C^* \circ \pi_{\mathsf{ddh}} \circ \pi^2_{\mathsf{enc}} \circ \pi^2_{\mathsf{bullet}})$	σ

Optimizations

randomness reuse

sn	$pk_s, C_s, pk_r, C_r, pk_a, C_a$	$\pi_{\mathsf{equal}} \circ (\pi^1_{\mathsf{enc}} \circ \pi^1_{\mathsf{bullet}}) \circ (C^* \circ \pi_{\mathsf{ddh}} \circ \pi^2_{\mathsf{enc}} \circ \pi^2_{\mathsf{bullet}})$	σ

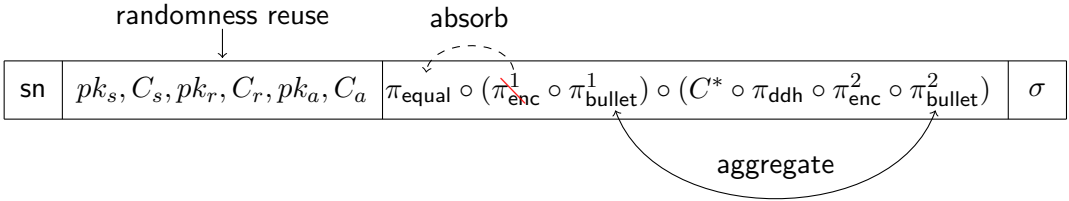
Randomness-Reusing

- original construction encrypts the same message v under pk_i $(i = \{s, r, a\}$ using independent random coins: $(pk_s, pk_s^{r_1}, g^{r_1}h^v, pk_r, pk_r^{r_2}, g^{r_2}h^v, pk_a, pk_a^{r_3}, g^{r_3}h^v)$
- twisted ElGamal is IND-CPA secure in 1-message/3-recipient setting

even when reusing randomness $\Rightarrow (pk_s, pk_s^r, pk_r, pk_r^r, pk_a, pk_a^r, \boxed{g^r h^v})$

Benefit: compact ctx size & simpler design of Σ_{enc}

Optimizations

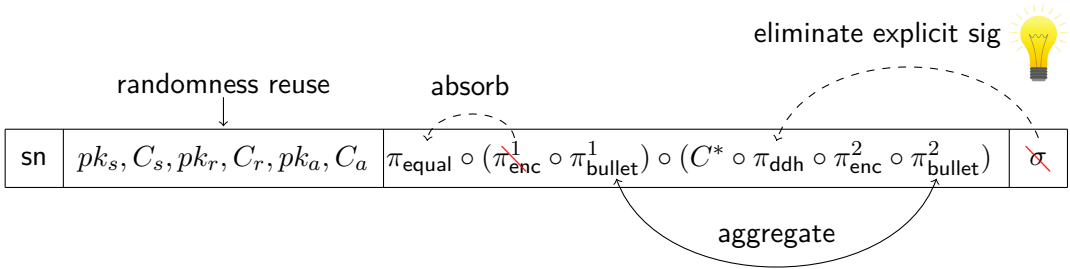


More Efficient Assembly of NIZK

- π_{enc} can be removed since π_{equal} already proves knowledge of C_s
- nice feature of twisted ElGamal $\Rightarrow$ two Bulletproofs can be generated and verified in aggregated mode $\rightsquigarrow$ reduce the size of range proof part by half

Benefit: further shrink the ctx size

Optimizations

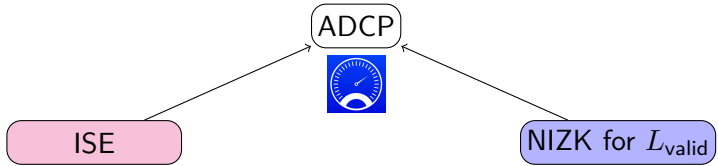


Eliminate Explicit Signature

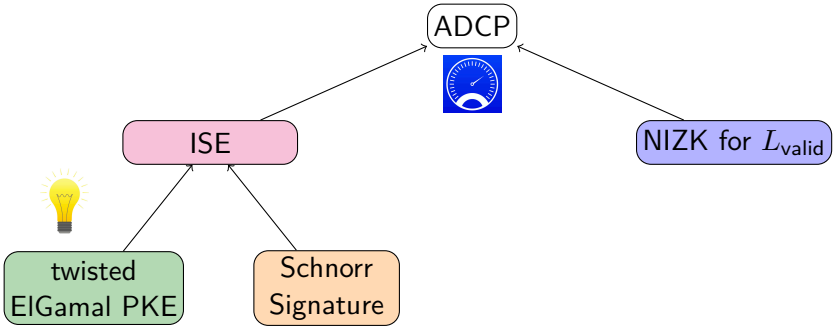
- Σ_{ddh} (3-move public-coin ZKPoK of sk_1) is a sub-protocol of NIZK for L_{solvent}
- apply the Fiat-Shamir transform by appending the rest part to hash input $\rightsquigarrow \pi_{\mathsf{ddh}}$ serves as both a proof of DDH tuple and a sEUF-CMA signature of ctx (jointly secure with twisted ElGamal)

Benefit: further shrink the ctx size & speed ctx generation/verification

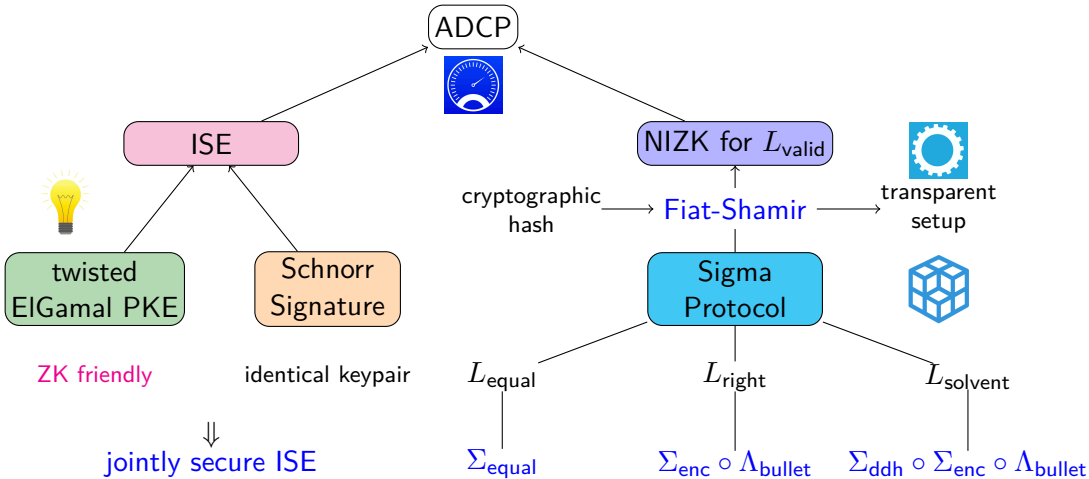
Recap of Efficient Instantiation



Recap of Efficient Instantiation



Recap of Efficient Instantiation



Outline

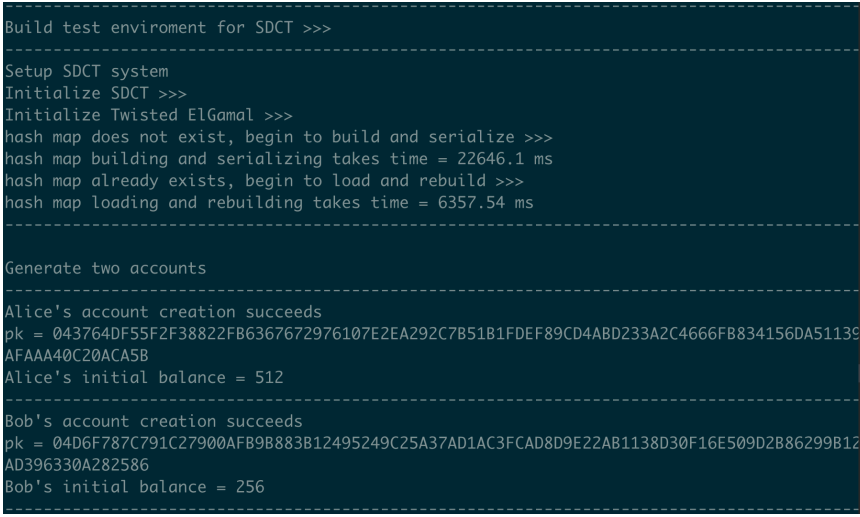
1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

Deploy as Cryptocurrency

Table: The computation and communication complexity of ADCP.

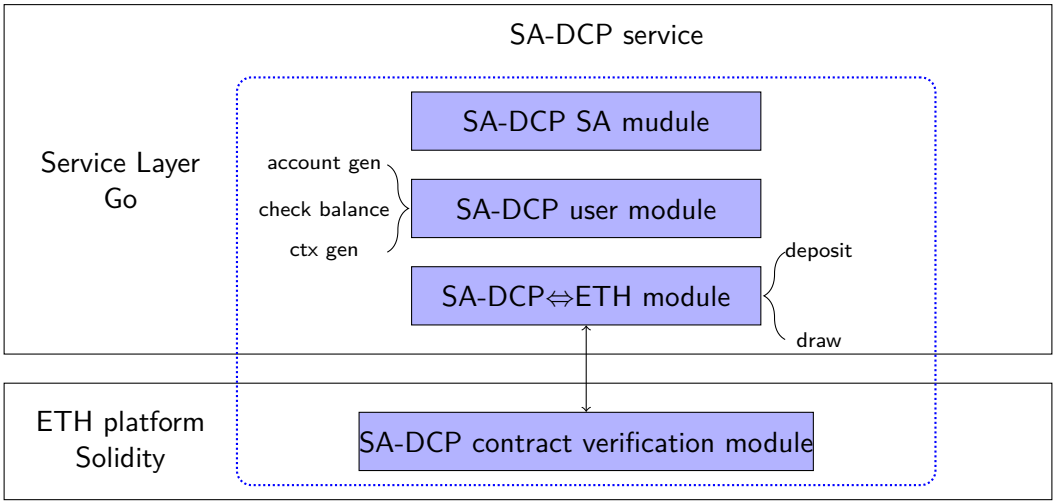
ADCP	ctx size		transaction cost (ms)	
	big-$\mathcal{O}$	bytes	generation	verify
transaction	$(2\log_2(\ell)+22)\|\mathbb{G}\|+11\|\mathbb{Z}_p\|$	1408	42	15
regulation	proof size		auditing cost (ms)	
	big-$\mathcal{O}$	bytes	generation	verify
limit policy	$(2\log_2(\ell)+4)\|\mathbb{G}\|+5\|\mathbb{Z}_p\|$	622	21.5	7.5
rate policy	$2\|\mathbb{G}\|+1\|\mathbb{Z}_p\|$	98	0.55	0.69
open policy	$2\|\mathbb{G}\|+1\|\mathbb{Z}_p\|$	98	0.26	0.42
supervision	opening $\leq$ 1ms			

- Set $v_{\max} = 2^{\ell} - 1$, where $\ell = 32$
- Choose EC curve prime256v1 (128 bit security), $|\mathbb{G}| = 33$ bytes, $|\mathbb{Z}_p| = 32$ bytes.
- MacBook Pro [Intel i7-4870HQ CPU (2.5GHz), 16GB of RAM]

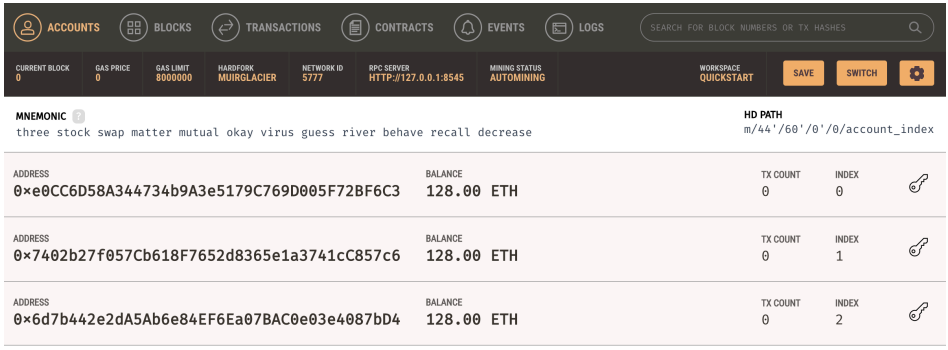

Build test enviroment for SDCT >>>
Setup SDCT system
Initialize SDCT >>>
Initialize Twisted ElGamal >>>
hash map does not exist, begin to build and serialize >>>
hash map building and serializing takes time = 22646.1 ms
hash map already exists, begin to load and rebuild >>>
hash map loading and rebuilding takes time = 6357.54 ms
Generate two accounts
Alice's account creation succeeds
pk = 043764DF55F2F38822FB6367672976107E2EA292C7B51B1FDEF89CD4ABD233A2C4666FB834156DA5113
AFAAA40C20ACA5B
Alice's initial balance = 512
Bob's account creation succeeds
pk = 04D6F787C791C27900AFB9B883B12495249C25A37AD1AC3FCAD8D9E22AB1138D30F16E509D2B86299B12
AD396330A282586
Bob's initial balance = 256
SDCT-CRYPTOCURRENCY
build
depends
bulletproofs
aggregate_bulletproof.hpp
innerproduct_proof.hpp
common
global.hpp
hash.hpp
print.hpp
routines.hpp
nizk
nizk_dlog_equality.hpp
nizk_plaintext_equality.hpp
nizk_plaintext_knowledge.hpp
sm
sm3hash.hpp
twisted_elgamal
calculate_dlog.hpp
twisted_elgamal.hpp
src
SDCT.hpp
test
CMakeLists.txt
LICENSE
README_cn.md
README_cn.pdf
README_en.md

Deploy as a Service

provide auditable confidential transaction service for ETH platform.



experimental result on ETH Ganache 2.4.0 $\rightsquigarrow$ SA-DCP service is practical




```
INFO[06-06|20:45:36] CTx transfer                                token=0x0000000000000000000000000000
amount=128 gas=2537804 tx=0xb8e319158cfd2996555a50f15e13069b811e85991d856a061abe00f4b5c8d4a3
--------------------------------------------------------------------------------
CTx transfer succeeds: Carol transfer 128 coins to Bob
Carol's current balance 0
Bob's current balance 256

INFO[06-06|20:45:43] CTx transfer                                token=0x0000000000000000000000000000
amount=128 gas=2499341 tx=0x58d522f75b0ab940ae0f47eb8c63bf85f89f9be5e4e035370cde0a45c3bc2416

CTx transfer succeeds: Bob transfer 128 coins to Alice
Bob's current balance 128
Alice's current balance 256
```

Outline

1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

Comparison to Related Works

Table: Comparison to other account-based DCP

Scheme	transparent setup	scalability	confidentiality	anonymity	regulation	supervision
zkLedger	✓+ DL	$O(n)$	?	✓	$O(m, \|f\|)$	✗
Zether	✓+ DL	$O(1)$	✓	✓	?	✗
ADCP	✓+ DL	$O(1)$	✓	✗	$O(\|f\|)$	✓

- n is the number of system users, m is the number of all transactions on the ledger
- zkLedger [NVV18]: (i) ctx size is linear of n, and n is fixed at the very beginning. (ii) confidentiality is questionable due to the use of correlated randomness; (iii) audit efficiency is linear of both m and $|f|$ due to anonymity
- Zether [BAZB20]: (i) possibly support audit when sacrificing anonymity; (ii) security of ZKP is hard to check

Summary

We propose a framework of ADCP from ISE and NIZK with formal security model and rigorous proof

- provide strong privacy and security guarantees for normal users
- provide handlers to conduct regulation and supervision for authority

We instantiate the ADCP by carefully designing and combining cryptographic primitives $\rightsquigarrow$ PGC

- transparent setup, security solely based on the DLOG assumption
- modular, simple and efficient

Highlights

- twisted ElGamal: efficient, homomorphic and zero-knowledge proof friendly $\rightsquigarrow$ a good alternative to ISO standard HE schemes: ElGamal and Paillier
- two proof gadgets: widely applicable in privacy-preserving scenarios, e.g. secure machine learning

History of This Work

2019.01: run out of ideas, begin to investigate cryptocurrency

2019.02: brain storming, solve a bunch of technical difficulties

- a simple twist $\Rightarrow$ twisted ElGamal

2019.03: finish a rush draft

2019.04-05: finish demo code based on MIRACL

2019.06-07: finish the security proofs of ZK part

2019.08-09: rewrite the demo code based on OpenSSL

2020.06: ESORICS 2020

2020.09-11: support supervision, winner of 1st FinCrypto competition

2021.09-10: rewrite the code with some new optimization tricks

Impact

CISC 2019

Challenge1

CISC 2019

Champion ($3000)

Team Mugiwara, Zhicong Huang from Alibaba Group

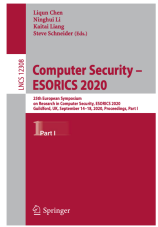
Runner Up ($1000)

Team Kunlun: Yu Chen from Ant Financial

3rd Place ($500)

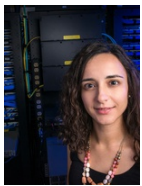
(Vacancy)

ESORICS 2020



首届金融密码杯 (3/205)

Foteini Baldimtsi @ GMU



We note that zkLedger [51] uses Pedersen commitments but overlooks the connection with twisted ElGamal. A proper use of twisted ElGamal in zkLedger can lead to optimizations as discussed in detail in Appendix D.

By using twisted ElGamal [25], MiniLedger is fully-compatible with Bulletproofs [17] which can further reduce its concrete storage requirements.

different public keys. PGC is one of the few works that recognizes the problem of efficient small *dlog* lookup tables, and while it highlights the greater efficiency of heuristic approaches like *kangaroo*, it still opts for Shanks to enable easy amortization for the time-space tradeoff and parallelization. In their proof of

Thanks for Your Attention!

Any Questions?

Reference I

[BAZB20] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. Zether: Towards privacy in a smart contract world. In *Financial Cryptography and Data Security - FC 2020*, volume 12059, pages 423–443. Springer, 2020.

[CCS08] Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 234–252. Springer, 2008.

[CKLR21] Geoffroy Couteau, Michael Klooß, Huang Lin, and Michael Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In *Advances in Cryptology - EUROCRYPT 2021*, volume 12698 of *LNCS*, pages 247–277. Springer, 2021.

[FMMO19] Prastudy Fauzi, Sarah Meiklejohn, Rebekah Mercer, and Claudio Orlandi. Quisquis: A new design for anonymous cryptocurrencies. In *Advances in Cryptology - ASIACRYPT 2019*, volume 11921 of *Lecture Notes in Computer Science*, pages 649–678. Springer, 2019.

[NVV18] Neha Narula, Willy Vasquez, and Madars Virza. zkledger: Privacy-preserving auditing for distributed ledgers. In *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018*, pages 65–80, 2018.

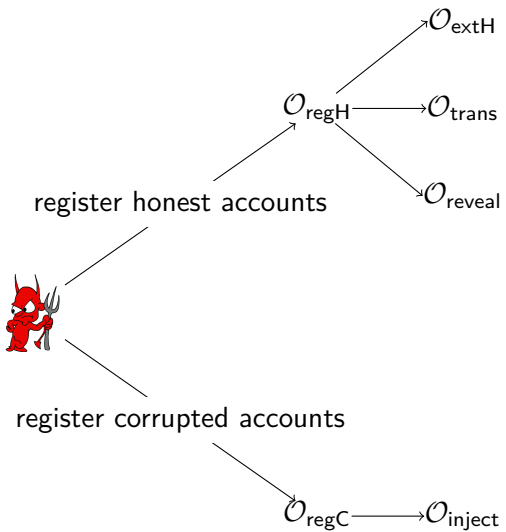
Outline

1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

Outline

1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

Formal Security Model (Oracles)



corrupt honest accounts

direct honest accounts to conduct ctx

ask honest accounts to reveal ctx

inject ctx from corrupted accounts

Formal Security Model: Authenticity

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c|l} \mathsf{VerifyCTx}(\mathsf{ctx}^*) = 1 \wedge & pp \leftarrow \mathsf{Setup}(\lambda); \\ pk_s^* \in T_{\mathsf{honest}} \wedge\ \mathsf{ctx}^* \notin T_{\mathsf{ctx}}(pk_s^*) & \mathsf{ctx}^* \leftarrow \mathcal{A}^{\mathcal{O}}(pp); \end{array}\right].$$

Formal Security Model: Confidentiality

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} pp \leftarrow \mathsf{Setup}(\lambda); \\ (state, pk_s^*, pk_r^*, v_0, v_1) \leftarrow \mathcal{A}_1^{\mathcal{O}}(pp); \\ \beta \xleftarrow{\mathsf{R}} \{0,1\}; \\ \mathsf{ctx}^* \leftarrow \mathsf{CreateCTx}(sk_s^*, pk_s^*, pk_r^*, v_\beta); \\ \beta' \leftarrow \mathcal{A}_2^{\mathcal{O}}(state, \mathsf{ctx}^*); \end{array}\right] - \frac{1}{2}.$$

To prevent trivial attacks, $\mathcal{A}$ is subject to the following restrictions:

1. pk_s^*, pk_r^* chosen by $\mathcal{A}$ are required to be honest accounts, and $\mathcal{A}$ is not allowed to make corrupt queries to either pk_s^* or pk_r^*;
2. $\mathcal{A}$ is not allowed to make reveal query to ctx^*.
3. let v_{sum} (with initial value 0) be the dynamic sum of the transfer amounts in $\mathcal{O}_{\mathsf{trans}}$ queries related to pk_s^* after ctx^*, both $\tilde{v}_s - v_0 - v_{\mathsf{sum}}$ and $\tilde{v}_s - v_1 - v_{\mathsf{sum}}$ must lie in $\mathcal{V}$.

Restrictions 1 and 2 prevents trivial attack by decryption, restrictions 3 prevent inferring β by testing whether overdraft happens.

Formal Security Model: Soundness

$$\mathsf{Adv}_{\mathcal{A}}(\lambda) = \Pr\left[\begin{array}{c|l} \mathsf{VerifyCTx}(\mathsf{ctx}^*) = 1 & pp \leftarrow \mathsf{Setup}(\lambda); \\ \wedge\ \mathsf{memo}^* \notin L_{\mathsf{valid}} & \mathsf{ctx}^* \leftarrow \mathcal{A}^{\mathcal{O}}(pp); \end{array}\right].$$

Here, $\mathsf{ctx}^* = (\mathsf{sn}^*, \mathsf{memo}^*, \mathsf{aux}^*)$.

Outline

1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

A Subtle Point: Key reuse vs. Key Separation

We employ PKE and SIG simutaneously to secure auditable DCP.

---

key separation
$(pk_1, sk_1), (pk_2, sk_2)$

Pros

- off-the-shelf & easy to analyze

Cons

- double key size
- tricky address derivation

key reuse
(pk, sk)

Pros

- greatly simplify DCP system
- more efficient

Cons

- case-tailored design

We choose Integrated Signature and Encryption (ISE): one keypair for both encryption and sign, while IND-CPA and EUF-CMA hold in the joint sense

Outline

1. Background
2. Framework of Auditable DCP System
3. An Efficient Instantiation: PGC
4. Experimental Results
5. Summary
6. Backup
   - Formal Security Model
   - Key Reuse vs. Key Separation
   - Generic Framework and Security Proof

Generic Construction of Auditable DCP: Building blocks

ISE $=$ (Setup, KeyGen, Sign, Verify, Enc, Dec)

- PKE component is additively homomorphic over $\mathbb{Z}_p$
- Fix pp, KeyGen naturally induces an $\mathcal{NP}$ relation:

$$\mathsf{R}_{\mathsf{key}} = \{(pk, sk) : \exists r \text{ s.t. } (pk, sk) = \mathsf{KeyGen}(pp; r)\}$$

NIZK $=$ (Setup, CRSGen, Prove, Verify)

- adaptive soundness
- adaptive ZK

Algorithms of Auditable DCP: 1/4

Setup(1^λ): generate pp for the auditable DCP system

- $pp_{\mathsf{ise}} \leftarrow \mathsf{ISE.Setup}(1^\lambda)$, $pp_{\mathsf{nizk}} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, $crs \leftarrow \mathsf{NIZK.CRSGen}(pp_{\mathsf{nizk}})$
- output $pp = (pp_{\mathsf{ise}}, pp_{\mathsf{nizk}}, crs)$, set $\mathcal{V} = [0, v_{\mathsf{max}}]$

CreateAcct($\tilde{v}$, sn): create an account

- $(pk, sk) \leftarrow \mathsf{ISE.KeyGen}(pp_{\mathsf{ise}})$, pk serves as account address
- $\tilde{C} \leftarrow \mathsf{ISE.Enc}(pk, \tilde{v}; r)$

RevealBalance($sk, \tilde{C}$): reveal the balance of an account

- $\tilde{m} \leftarrow \mathsf{ISE.Dec}(sk, \tilde{C})$

Algorithms of Auditable DCP: 2/4

CreateCTx(sk_s, pk_s, v, pk_r): transfer v coins from account pk_s to account pk_r.

- $C_s \leftarrow \mathsf{ISE.Enc}(pk_s, v; r_1)$, $C_r \leftarrow \mathsf{ISE.Enc}(pk_r, v; r_2)$, $\mathsf{memo} = (pk_s, pk_r, C_s, C_r)$.
- run NIZK.Prove with witness (sk_s, r_1, r_2, v) to generate a proof π_{valid} for $\mathsf{memo} = (pk_s, pk_r, C_s, C_r) \in L_{\mathsf{valid}} \mapsto L_{\mathsf{equal}} \wedge L_{\mathsf{right}} \wedge L_{\mathsf{solvent}}$

$$L_{\mathsf{equal}} = \{(pk_s, pk_r, C_s, C_r) \mid \exists r_1, r_2, v \text{ s.t. } C_s = \mathsf{ISE.Enc}(pk_s, v; r_1) \wedge C_r = \mathsf{ISE.Enc}(pk_r, v; r_2)\}$$
$$L_{\mathsf{right}} = \{(pk_s, C_s) \mid \exists r_1, v \text{ s.t. } C_s = \mathsf{ISE.Enc}(pk_s, v; r_1) \wedge v \in \mathcal{V}\}$$
$$L_{\mathsf{solvent}} = \{(pk_s, \tilde{C}_s, C_s) \mid \exists sk_1 \text{ s.t. } (pk_s, sk_s) \in \mathsf{R_{key}} \wedge \mathsf{ISE.Dec}(sk_s, \tilde{C}_s - C_s) \in \mathcal{V}\}$$

- $\sigma \leftarrow \mathsf{ISE.Sign}(sk_s, (\mathsf{sn}, \mathsf{memo}, \pi_{\mathsf{valid}}))$
- output $\mathsf{ctx} = (\mathsf{sn}, \mathsf{memo}, \pi_{\mathsf{valid}}, \sigma)$.

Algorithms of Auditable DCP: 3/4

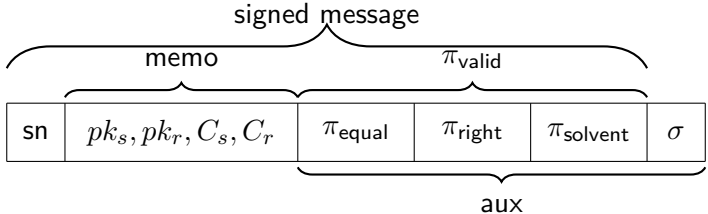


Figure: Data structure of confidential transaction.

VerifyCTx(ctx): check if ctx is valid.

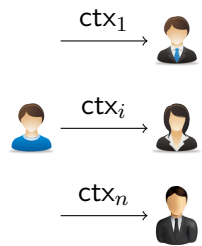
- parse ctx $= (\text{sn}, \text{memo}, \pi_{\text{valid}}, \sigma)$, memo $= (pk_s, pk_r, C_s, C_r)$:
  1. check if sn is a fresh serial number of pk_s (inspect the blockchain);
  2. check if ISE.Verify$(pk_s, (\text{sn}, \text{memo}, \pi_{\text{valid}}), \sigma) = 1$;
  3. check if NIZK.Verify$(crs, \text{memo}, \pi_{\text{valid}}) = 1$.
- ctx is recorded on the ledger if validity test passes or discarded otherwise.

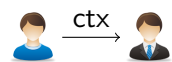
Update(ctx): sender updates his balance $\tilde{C}_s = \tilde{C}_s - C_s$ and increments sn, receiver updates his balance $\tilde{C}_r = \tilde{C}_r + C_r$.

Algorithms of Auditable DCP: 4/4

JustifyCTx$(pk, sk, \{\mathsf{ctx}_i\}_{i=1}^n, f)$: user pk runs NIZK.Prove with witness sk to generate a zero-knowledge proof π_f for $f(\{\mathsf{ctx}_i\}_{i=1}^n) = 1$.

$f_{\mathsf{limit}} : \sum_{i=1}^n v_i < \ell$	$f_{\mathsf{rate}} : v_1/v_2 = \rho$	$f_{\mathsf{open}} : v = v^*$
anti-money laundering	tax payment	selective disclosure





AuditCTx$(pk, \{\mathsf{ctx}_i\}_{i=1}^n, f, \pi_f)$: auditor runs NIZK.Verify to check if π_f is valid.

Security Proof

Theorem: Assuming the security of ISE and NIZK, our CTx framework is secure.

- security of ISE's signature component $\Rightarrow$ authenticity
- security of ISE's PKE component + adaptive ZK of NIZK $\Rightarrow$ confidentiality
- adaptive soundness of NIZK $\Rightarrow$ soundness