YOLO YOSO
Fast and Simple Encryption and Secret Sharing in the YOSO Model

Ignacio Cascudo	IMDEA Software Institute, Madrid
Bernardo David	IT University of Copenhagen
Lydia Garms	Keyless Technologies Limited, London
Anders Konring	IT University of Copenhagen

The YOSO model *(Gentry et al. Crypto 21)*

- Traditional Multiparty Computation Model:
  - Fixed set of parties.
  - Several rounds, each party send messages to others.

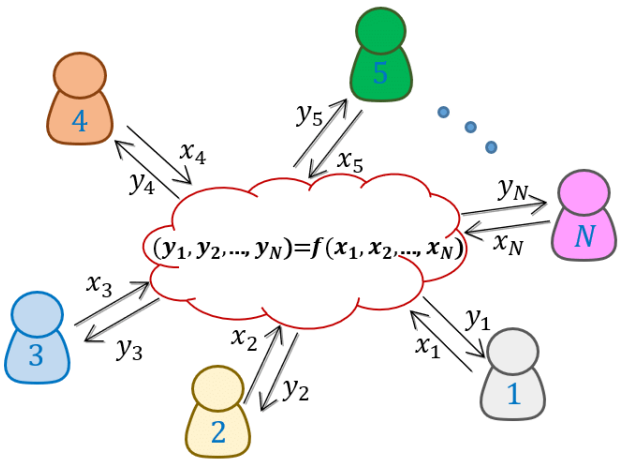
The YOSO model *(Gentry et al. Crypto 21)*

- Traditional Multiparty Computation Model:
  - Fixed set of parties.
  - Several rounds, each party send messages to others.

- YOSO (You Only Speak Once) MPC Model:
  - Very large pool of parties.
  - Each round is executed by a committee.
  - Each party speaks at most once. Identity of party not known until they speak.
  - Difficulties corruption by adversaries.

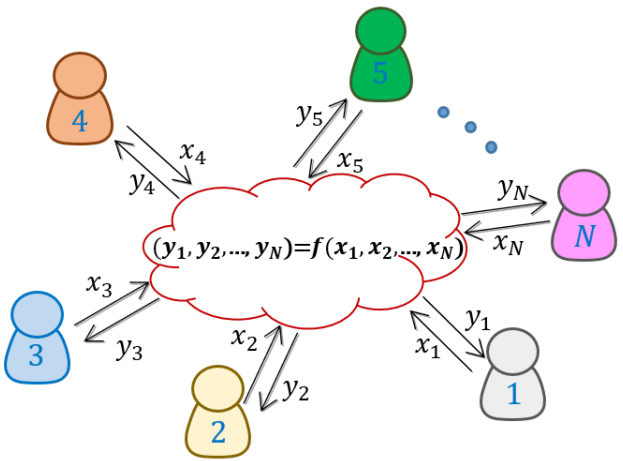
From PKE to YOSO

- Traditional MPC:
  - Ideal secure channels
  - Instantiation using public key encryption.

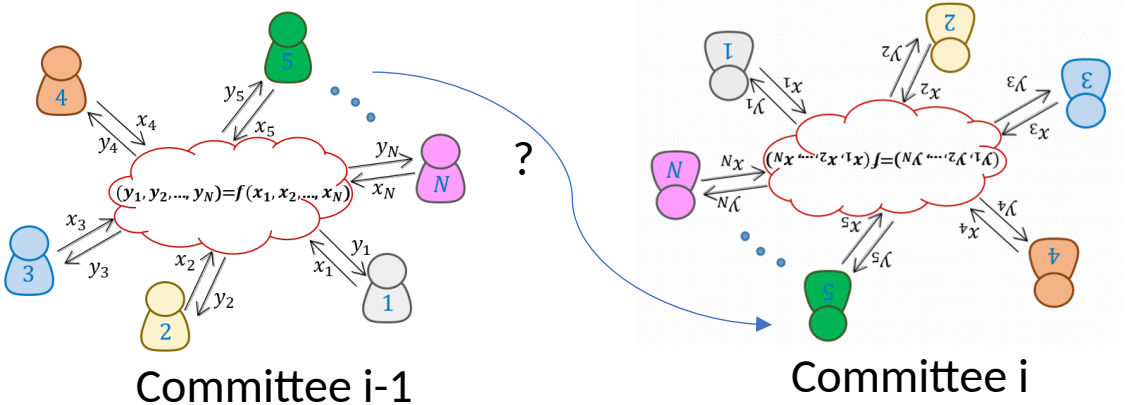


From PKE to YOSO

- Traditional MPC:
  - Ideal secure channels
  - Instantiation using public key encryption.



- YOSO MPC :
  - Every round is executed by a different anonymous committee.
  - How to encrypt towards an unknown party?



Previous Approaches

- "**Can a Blockchain keep a secret?**" *(Benhamouda et al., TCC 20)*: committee holds secret shared message, <25% corruption tolerated.

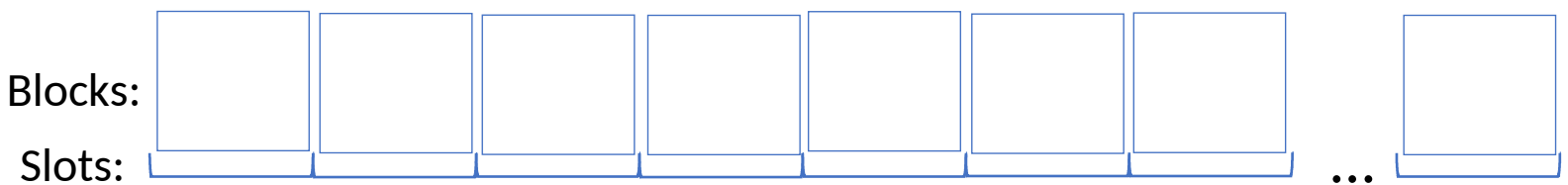
- **YOSO MPC**: idealized model of role assignment and future broadcast

- **Random-index PIR** *(Gentry et al. TCC 21)*: parties retrieve random anonymous public keys

Encryption to the Future

(*Campanelli et al., Asiacrypt 22*, Friday talk)

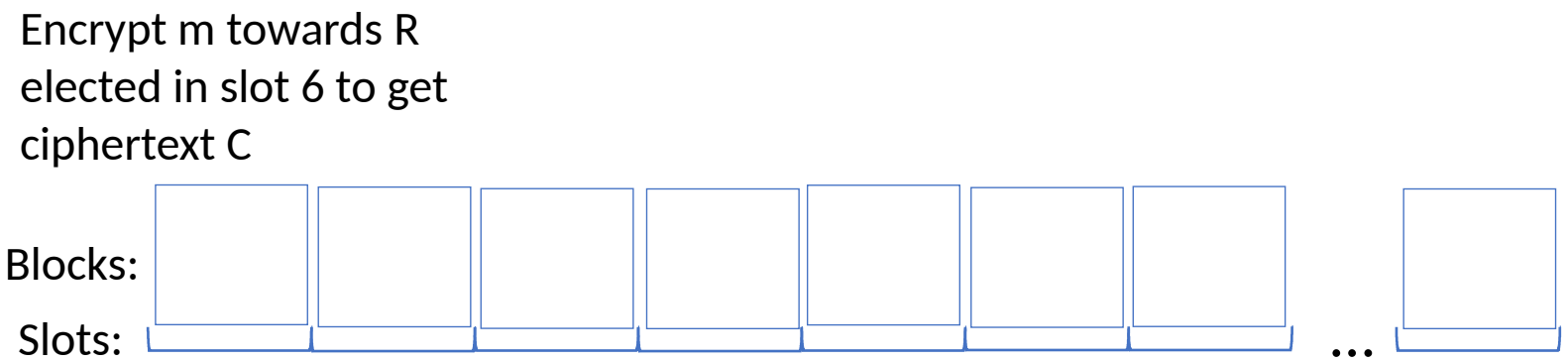
- Time is divided into slots, kept by a blockchain.



Encryption to the Future

(*Campanelli et al., Asiacrypt 22*, Friday talk)

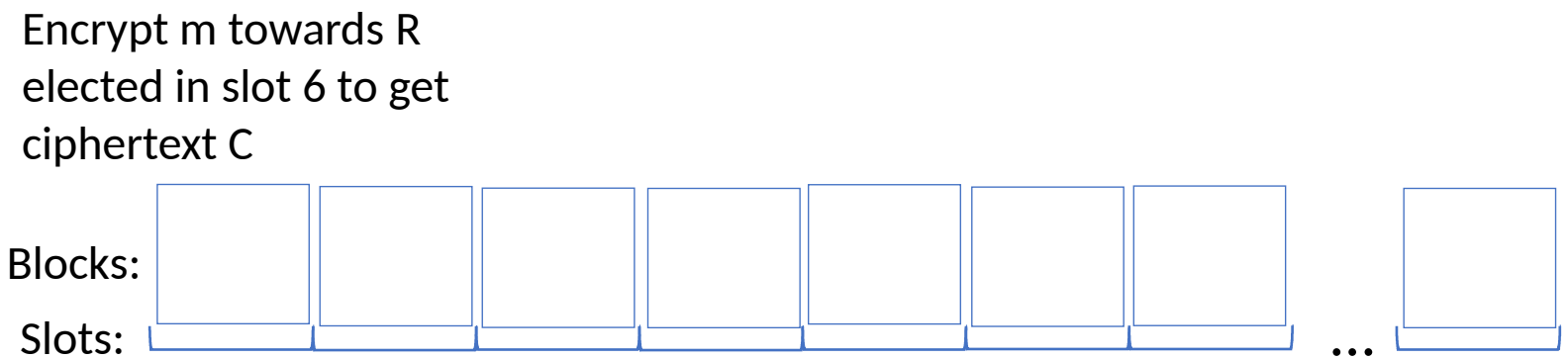
- Time is divided into slots, kept by a blockchain.
- Encrypt towards a "role" **R** related to a future slot **sl** given current chain.



Encryption to the Future

(*Campanelli et al., Asiacrypt 22*, Friday talk)

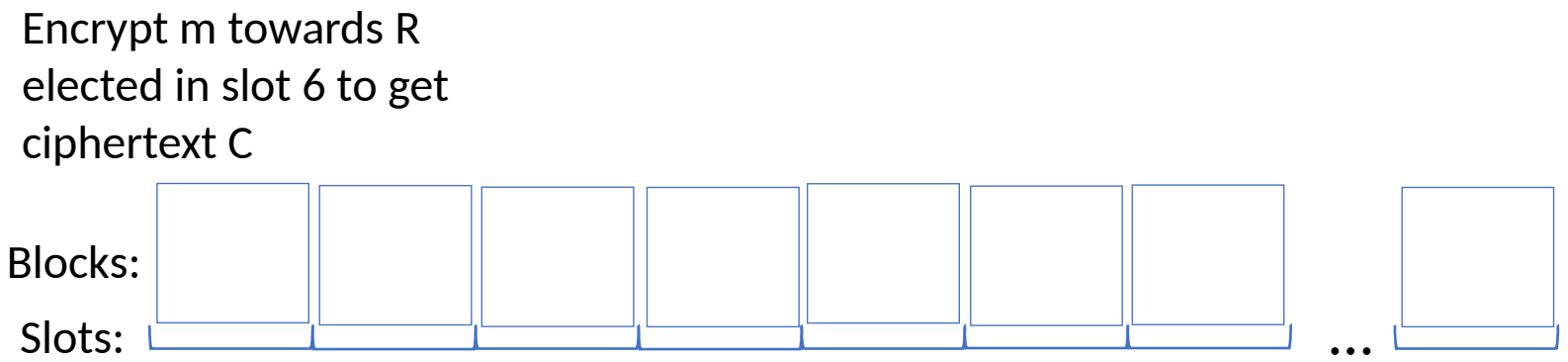
- Time is divided into slots, kept by a blockchain.
- Encrypt towards a "role" **R** related to a future slot **sl** given current chain.
- Lottery predicate checks if owner of **sk** will perform role **R** in slot **sl** on chain **B**: **Lottery(B,sl,R,sk)={0,1}**



Encryption to the Future

(*Campanelli et al., Asiacrypt 22*, Friday talk)

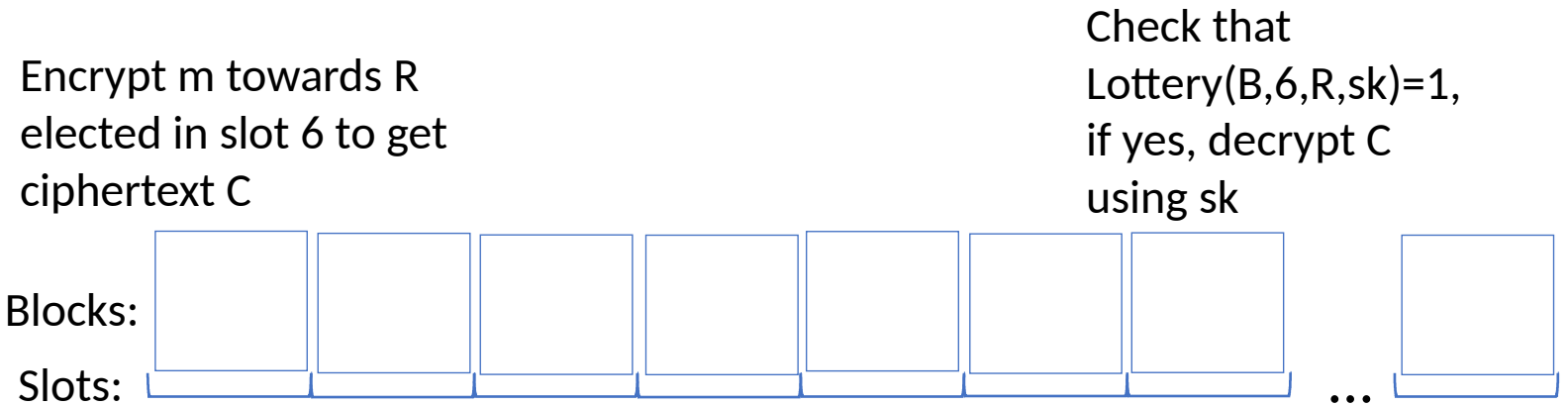
- Time is divided into slots, kept by a blockchain.
- Encrypt towards a "role" **R** related to a future slot **sl** given current chain.
- Lottery predicate checks if owner of **sk** will perform role **R** in slot **sl** on chain **B**: **Lottery(B,sl,R,sk)={0,1}**
- When blockchain reaches slot sl, owner of sk with (B,sl,R,sk)=1 can decrypt.



Encryption to the Future

(*Campanelli et al., Asiacrypt 22*, Friday talk)

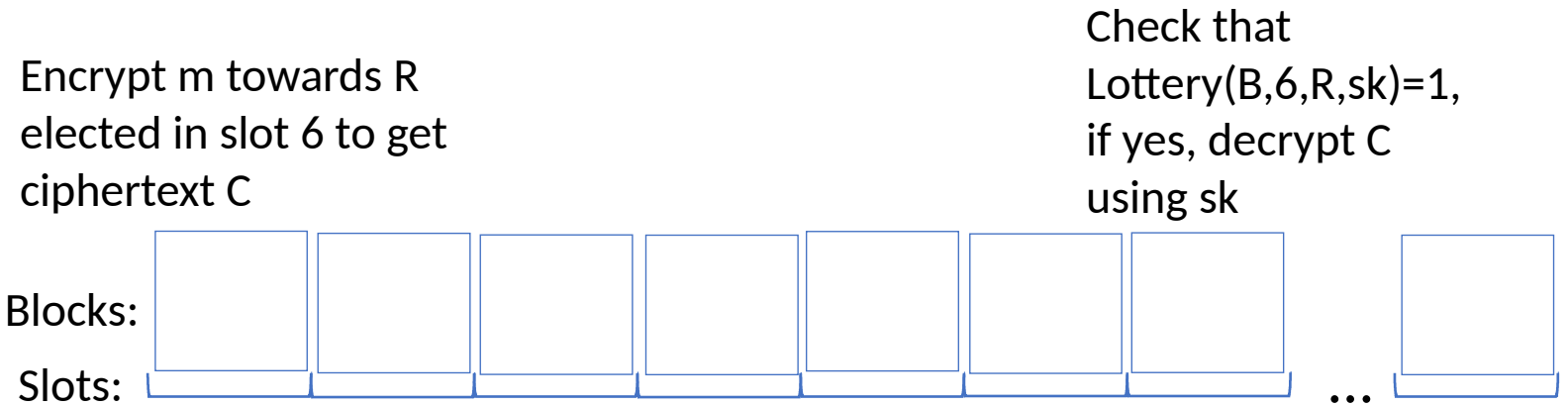
- Time is divided into slots, kept by a blockchain.
- Encrypt towards a "role" **R** related to a future slot **sl** given current chain.
- Lottery predicate checks if owner of **sk** will perform role **R** in slot **sl** on chain **B**: **Lottery(B,sl,R,sk)={0,1}**
- When blockchain reaches slot sl, owner of sk with (B,sl,R,sk)=1 can decrypt.



Encryption to the Future

(*Campanelli et al., Asiacrypt 22*, Friday talk)

- Time is divided into slots, kept by a blockchain.
- Encrypt towards a "role" **R** related to a future slot **sl** given current chain.
- Lottery predicate checks if owner of **sk** will perform role **R** in slot **sl** on chain **B**: **Lottery(B,sl,R,sk)={0,1}**
- When blockchain reaches slot sl, owner of sk with (B,sl,R,sk)=1 can decrypt.



Encryption to Current Winner

- The party with role R is determined at the time of encryption.

Encryption to Current Winner

- The party with role R is determined at the time of encryption.

- Simpler case
  - Solutions for ETF in Campanelli et al: O(n) communication
  - Solutions for ECW in this paper: O(1)

Encryption to Current Winner

- The party with role R is determined at the time of encryption.
- Simpler case
  - Solutions for ETF in Campanelli et al: O(n) communication
  - Solutions for ECW in this paper: O(1)
- Transformation of ECW into full blown ETF using IBE and a committee that only holds shares of an IBE master secret key

Our Contributions

Our Contributions

- Concretely efficient construction of ECW using **shuffling**

Our Contributions

- Concretely efficient construction of ECW using **shuffling**
- Given an anonymous broadcast channel we get non-interactive ECW with **ciphertext length independent** from number of parties

Our Contributions

- Concretely efficient construction of ECW using **shuffling**
- Given an anonymous broadcast channel we get non-interactive ECW with **ciphertext length independent** from number of parties
- Publicly Verifiable Secret Sharing (PVSS) scheme compatible with our ECW.

Our Contributions

- Concretely efficient construction of ECW using **shuffling**
- Given an anonymous broadcast channel we get non-interactive ECW with **ciphertext length independent** from number of parties
- Publicly Verifiable Secret Sharing (PVSS) scheme compatible with our ECW.
- DDH based instantiation of ECW+PVSS, sharing proof **independent from number of parties.**

Our Contributions

- Concretely efficient construction of ECW using **shuffling**
- Given an anonymous broadcast channel we get non-interactive ECW with **ciphertext length independent** from number of parties
- Publicly Verifiable Secret Sharing (PVSS) scheme compatible with our ECW.
- DDH based instantiation of ECW+PVSS, sharing proof **independent from number of parties.**
- PVSS resharing protocols

ECW through Shuffling

ECW through Shuffling

- We propose a simple approach:

ECW through Shuffling

- We propose a simple approach:
  - Each party selects an anonymous key pair.

ECW through Shuffling

- We propose a simple approach:
  - Each party selects an anonymous key pair.
  - Concretely, each party inputs their PK into a mix-net. The resulting list of PKs is published in the blockchain.

ECW through Shuffling

- We propose a simple approach:
  - Each party selects an anonymous key pair.
  - Concretely, each party inputs their PK into a mix-net. The resulting list of PKs is published in the blockchain.
  - Lottery will select an unused key in the list and encryptor will be encrypting under that PK

ECW through Shuffling

- We propose a simple approach:
  - Each party selects an anonymous key pair.
  - Concretely, each party inputs their PK into a mix-net. The resulting list of PKs is published in the blockchain.
  - Lottery will select an unused key in the list and encryptor will be encrypting under that PK
- More ellaborated strategy using Camenish-Lysyanskaya signatures: preserves some anonimity even after speaking.

ECW through Shuffling

- We propose a simple approach:
  - Each party selects an anonymous key pair.
  - Concretely, each party inputs their PK into a mix-net. The resulting list of PKs is published in the blockchain.
  - Lottery will select an unused key in the list and encryptor will be encrypting under that PK
- More ellaborated strategy using Camenish-Lysyanskaya signatures: preserves some anonimity even after speaking.
- This is good enough for ECW, but how about secret sharing to a committee? How do we prove consistency between shares?

Publicly Verifiable Secret Sharing (PVSS)

Publicly Verifiable Secret Sharing (PVSS)

- Secret sharing with publicly verifiable proofs of sharing correctness.

Publicly Verifiable Secret Sharing (PVSS)

- Secret sharing with publicly verifiable proofs of sharing correctness.
- Applications such as distributed randomness generation

Publicly Verifiable Secret Sharing (PVSS)

- Secret sharing with publicly verifiable proofs of sharing correctness.
- Applications such as distributed randomness generation
- But also: Key property for keeping secrets in a blockchain

Publicly Verifiable Secret Sharing (PVSS)

- Secret sharing with publicly verifiable proofs of sharing correctness.
- Applications such as distributed randomness generation
- But also: Key property for keeping secrets in a blockchain

Publicly Verifiable Secret Sharing (PVSS)

- Secret sharing with publicly verifiable proofs of sharing correctness.
- Applications such as distributed randomness generation
- But also: Key property for keeping secrets in a blockchain

Publicly Verifiable Secret Sharing (PVSS)

- Secret sharing with publicly verifiable proofs of sharing correctness.
- Applications such as distributed randomness generation
- But also: Key property for keeping secrets in a blockchain

PVSS Constructions

- We present two constructions of PVSS:
  - HEPVSS: Generic PVSS from a $\mathbf{Z}_p$-Linearly Homomorphic Encryption (LHE) scheme
  - DHPVSS: DL-based PVSS with **constant size proof of sharing correctness.**

$\mathbf{Z}_p$-Linearly Homomorphic Encryption

$\mathbf{Z}_p$-Linearly Homomorphic Encryption

- Encryption scheme where:
  - Plaintext, randomness, ciphertext spaces are $\mathbf{Z}_p$-linear spaces (e.g. groups of order p)
  - $E(m_1 +_P m_2; r_1 +_R r_2) = E(m_1; r_1) +_C E(m_2; r_2)$
  - Allows for simple “Schnorr-like” PoK of plaintext

$\mathbf{Z}_p$-Linearly Homomorphic Encryption

- Encryption scheme where:
  - Plaintext, randomness, ciphertext spaces are $\mathbf{Z}_p$-linear spaces (e.g. groups of order p)
  - $E(m_1 +_P m_2; r_1 +_R r_2) = E(m_1; r_1) +_C E(m_2; r_2)$
  - Allows for simple “Schnorr-like” PoK of plaintext
- We also want proof of correct decryption. Either:
  - Randomness recoverable
  - Decryption linear in secret key

$\mathbf{Z}_p$-Linearly Homomorphic Encryption

- Encryption scheme where:
  - Plaintext, randomness, ciphertext spaces are $\mathbf{Z}_p$-linear spaces (e.g. groups of order p)
  - $E(m_1 +_P m_2; r_1 +_R r_2) = E(m_1; r_1) +_C E(m_2; r_2)$
  - Allows for simple “Schnorr-like” PoK of plaintext
- We also want proof of correct decryption. Either:
  - Randomness recoverable
  - Decryption linear in secret key
- Satisfied by El Gamal encryption.

HEPVSS

HEPVSS

- Dealer creates sharing of s: $(\sigma_1,...,\sigma_n)$
- Dealer publishes $c_i = Enc_{pk_i}(\sigma_i, r_i)$

HEPVSS

- Dealer creates sharing of s: $(\sigma_1,...,\sigma_n)$
- Dealer publishes $c_i=Enc_{pk_i}(\sigma_i,r_i)$
- Dealer publishes encryption of sharing $(\sigma'_1,...,\sigma'_n)$ of random secret $c'_i=Enc_{pk_i}(\sigma'_i,r'_i)$.

HEPVSS

- Dealer creates sharing of s: $(\sigma_1,...,\sigma_n)$
- Dealer publishes $c_i=Enc_{pk_i}(\sigma_i,r_i)$
- Dealer publishes encryption of sharing $(\sigma'_1,...,\sigma'_n)$ of random secret $c'_i=Enc_{pk_i}(\sigma'_i,r'_i)$.
- Open $z_i=\sigma'_i+e\cdot\sigma_i$, and $t_i=r'_i+e\cdot r_i$, Fiat-Shamir challenge e.

HEPVSS

- Dealer creates sharing of s: $(\sigma_1,...,\sigma_n)$
- Dealer publishes $c_i=Enc_{pk_i}(\sigma_i,r_i)$
- Dealer publishes encryption of sharing $(\sigma'_1,...,\sigma'_n)$ of random secret $c'_i=Enc_{pk_i}(\sigma'_i,r'_i)$.
- Open $z_i=\sigma'_i+e\cdot\sigma_i$ and $t_i=r'_i+e\cdot r_i$ Fiat-Shamir challenge e.
- Verify $c'_i+e\cdot c_i= Enc_{pk_i}(z_i,t_i)$ and that $(z_1,...,z_n)$ is a correct sharing.

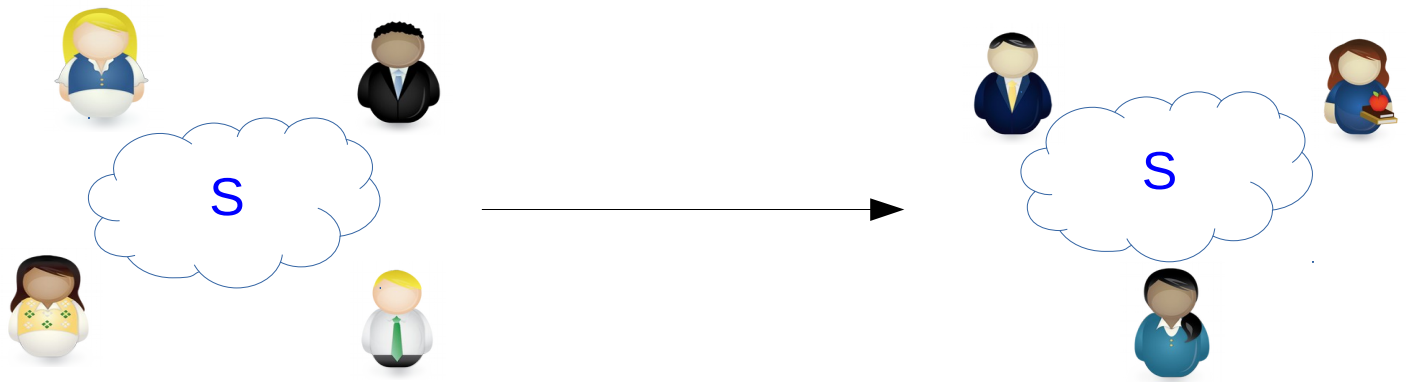
HEPVSS

- Dealer creates sharing of s: $(\sigma_1,...,\sigma_n)$
- Dealer publishes $c_i=Enc_{pk_i}(\sigma_i,r_i)$
- Dealer publishes encryption of sharing $(\sigma'_1,...,\sigma'_n)$ of random secret $c'_i=Enc_{pk_i}(\sigma'_i,r'_i)$.
- Open $z_i=\sigma'_i+e\cdot\sigma_i$ and $t_i=r'_i+e\cdot r_i$ Fiat-Shamir challenge e.
- Verify $c'_i+e\cdot c_i= Enc_{pk_i}(z_i,t_i)$ and that $(z_1,...,z_n)$ is a correct sharing.

Secure (indistinguishability of secrets) if Enc IND-CPA.

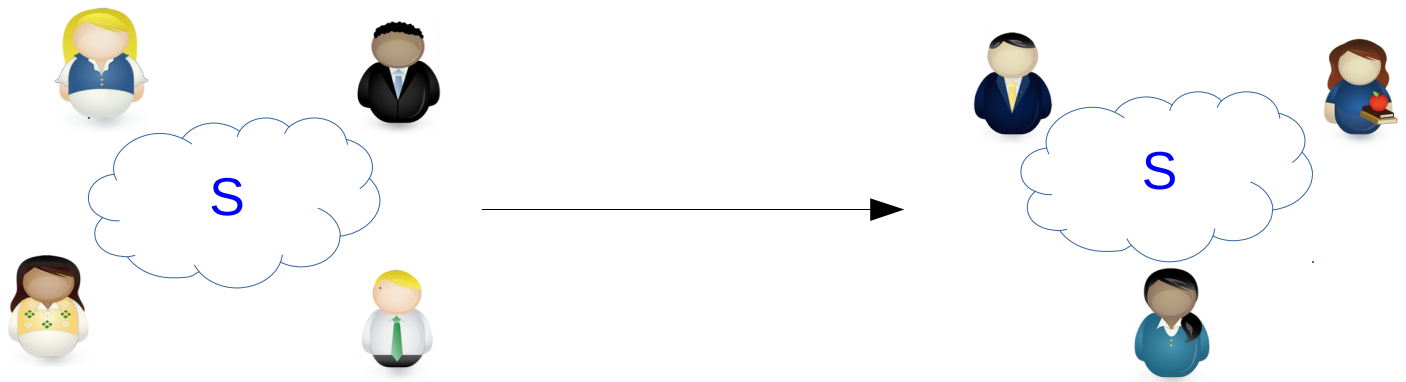
Resharing and Reconstruction

- Resharing to next committee with publicly verifiable proofs of correctness.
- Similar to PVSS, but in addition parties prove in ZK that the secret is their share. Uses proof of correct decryption.



Resharing and Reconstruction

- Resharing to next committee with publicly verifiable proofs of correctness.
- Similar to PVSS, but in addition parties prove in ZK that the secret is their share. Uses proof of correct decryption.
- New number of parties n and threshold t do not need to be the same.
- In particular, this implies proofs of correct reconstruction (n=1, t=0).



DHPVSS

DHPVSS

- We present a DL-based PVSS with **constant size overhead**

DHPVSS

- We present a DL-based PVSS with **constant size overhead**
- **First one as far as we know**

DHPVSS

- We present a DL-based PVSS with **constant size overhead**
- **First one as far as we know**
- Secret is an element S of group $\mathcal{G}$=<G> of order p, where DDH is hard.

DHPVSS

- We present a DL-based PVSS with **constant size overhead**
- **First one as far as we know**
- Secret is an element S of group $\mathcal{G}$=<G> of order p, where DDH is hard.
- Dealer publishes:
  - n encrypted shares: each 1 element in $\mathcal{G}$
  - Correctness proof: 2 $\mathbf{Z}_p$ elements

DHPVSS - Ideas 1

DHPVSS - Ideas 1

- In previous DL-based PVSS share-receivers have key pairs (sk_i, $PK_i = sk_i \cdot G$)

DHPVSS - Ideas 1

- In previous DL-based PVSS share-receivers have key pairs (sk_i, $PK_i = sk_i \cdot G$)
- **New**: Dealer will also have a key pair (sk_D, $PK_D = sk_D \cdot G$)

DHPVSS - Ideas 1

- In previous DL-based PVSS share-receivers have key pairs (sk_i, $PK_i = sk_i \cdot G$)
- **New**: Dealer will also have a key pair (sk_D, $PK_D = sk_D \cdot G$)
- Shamir shares of $S = s \cdot G$: $A_i = f(i) \cdot G$, where deg $f(x) \leq t$, $f(0) = S$

DHPVSS - Ideas 1

- In previous DL-based PVSS share-receivers have key pairs (sk_i, $PK_i=sk_i \cdot G$)
- **New**: Dealer will also have a key pair (sk_D, $PK_D=sk_D \cdot G$)
- Shamir shares of $S=s \cdot G$: $A_i=f(i) \cdot G$, where deg $f(x) \leq t$, $f(0)=S$
  - Dealer does not need to know s:
    - sample h, where deg $h(x) \leq t$, $h(0)=s$
    - set i-th share as $A_i = S + h(i) \cdot G$ *[Implicitely $f(x):=h(x)+S$]*

DHPVSS - Ideas 1

- In previous DL-based PVSS share-receivers have key pairs (sk_i, $PK_i=sk_i \cdot G$)
- **New**: Dealer will also have a key pair (sk_D, $PK_D=sk_D \cdot G$)
- Shamir shares of $S=s \cdot G$: $A_i=f(i) \cdot G$, where deg $f(x) \leq t$, $f(0)=S$
  - Dealer does not need to know s:
    - sample h, where deg $h(x) \leq t$, $h(0)=s$
    - set i-th share as $A_i = S + h(i) \cdot G$ *[Implicitely $f(x):=h(x)+S$]*
- A_i encrypted as $C_i=A_i+ sk_D \cdot PK$

DHPVSS - Ideas 2

DHPVSS - Ideas 2

- “SCRAPE check” *(Cascudo, David, ACNS 17)*

DHPVSS - Ideas 2

- “SCRAPE check” *(Cascudo, David, ACNS 17)*
  - $\mathcal{C}$ ={(f(1),...,f(n)): deg f ≤ t} is a linear (Reed-Solomon) code.
  - It has a dual code $\mathcal{D}$.

DHPVSS - Ideas 2

- “SCRAPE check” *(Cascudo, David, ACNS 17)*
  - $\mathcal{C}$ ={(f(1),...,f(n)): deg f ≤ t} is a linear (Reed-Solomon) code.
  - It has a dual code $\mathcal{D}$.
  - Let **c**=(c_1,...,c_n) in $(\mathbf{Z}_p)^n$. Sample **d**=(d_1,...,d_n) from $\mathcal{D}$. Then:
    - If **c** in $\mathcal{C}$, then $\Sigma\, d_i\, c_i = 0$
    - If **c** not in $\mathcal{C}$, then $\Sigma\, d_i\, c_i = 0$ with probability 1/p

DHPVSS - Ideas 2

- "SCRAPE check" *(Cascudo, David, ACNS 17)*
  - $\mathcal{C}$ ={(f(1),...,f(n)): deg f ≤ t} is a linear (Reed-Solomon) code.
  - It has a dual code $\mathcal{D}$.
  - Let **c**=(c_1,...,c_n) in $(\mathbf{Z}_p)^n$. Sample **d**=(d_1,...,d_n) from $\mathcal{D}$. Then:
    - If **c** in $\mathcal{C}$, then $\Sigma\, d_i\, c_i = 0$
    - If **c** not in $\mathcal{C}$, then $\Sigma\, d_i\, c_i = 0$ with probability 1/p

- Extends to group $\mathcal{G}$=<G>:
  - Given C_1,...,C_n in $\langle G \rangle^n$. Sample **d**=(d_1,...,d_n) from $\mathcal{D}$.
    - If $C_i = f(i) \cdot G$, deg f ≤ t, i=1,...,n, then $\Sigma\, d_i \cdot C_i = 0$
    - If not, $\Sigma\, d_i \cdot C_i = 0$ with probability 1/p

DHPVSS in detail

$PK_D = sk_D \cdot G$

S

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$ $C_i = f(i) \cdot G + sk_i \cdot PK_D$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$ $C_i = f(i) \cdot G + sk_i \cdot PK_D$ $\xrightarrow{sk_i}$ $f(i) \cdot G$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$ $C_i = f(i) \cdot G + sk_i \cdot PK_D$ $\xrightarrow{sk_i}$ $f(i) \cdot G$

Proof: $(d_1, \ldots, d_n) = H((C_i), (PK_i))$ publicly computable dual codeword (H random oracle)

$$\sum d_i C_i = \sum d_i f(i) \cdot G + \sum d_i sk_D \cdot PK_i$$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$ $C_i = f(i) \cdot G + sk_i \cdot PK_D$ $\xrightarrow{sk_i}$ $f(i) \cdot G$

Proof: $(d_1, \ldots, d_n) = H((C_i), (PK_i))$ publicly computable dual codeword (H random oracle)

$$\sum d_i C_i = \cancel{\sum d_i f(i) \cdot G}^{\,0} + \sum d_i sk_D \cdot PK_i$$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$ $C_i = f(i) \cdot G + sk_i \cdot PK_D$ $\xrightarrow{sk_i}$ $f(i) \cdot G$

Proof: $(d_1, \ldots, d_n) = H((C_i), (PK_i))$ publicly computable dual codeword (H random oracle)

$$\sum d_i C_i = \sum d_i f(i) \cdot G + \sum d_i sk_D \cdot PK_i = sk_D \sum d_i \cdot PK_i$$

($\sum d_i f(i) \cdot G \to 0$)

DHPVSS in detail

$PK_D = sk_D \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

S $\longrightarrow$ $f(i) \cdot G$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$ $C_i = f(i) \cdot G + sk_i \cdot PK_D$ $\xrightarrow{sk_i}$ $f(i) \cdot G$

Proof:

$(d_1, \ldots, d_n) = H((C_i), (PK_i))$ publicly computable dual codeword (H random oracle)

$$V = \sum d_i C_i = \sum d_i f(i) \cdot G + \sum d_i sk_D \cdot PK_i = sk_D \sum d_i \cdot PK_i = sk_D \cdot U$$

($\sum d_i f(i) \cdot G \to 0$)

DHPVSS in detail

$PK_D = sk_D \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

S $\longrightarrow$ $f(i) \cdot G$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$ $C_i = f(i) \cdot G + sk_i \cdot PK_D$ $\xrightarrow{sk_i}$ $f(i) \cdot G$

Proof: $(d_1, \ldots, d_n) = H((C_i), (PK_i))$ publicly computable dual codeword (H random oracle)

$$V = \sum d_i C_i = \sum d_i f(i) \cdot G + \sum d_i sk_D \cdot PK_i = sk_D \sum d_i \cdot PK_i = sk_D \cdot U$$

($\sum d_i f(i) \cdot G \to 0$)

π PoK: $sk_D = DL_G(PK_D) = DL_U(V)$

DHPVSS in detail

$PK_D = sk_D \cdot G$

S $\longrightarrow$ $f(i) \cdot G$

$PK_i = sk_i \cdot G, i = 1, \ldots, n$

$C_i = f(i) \cdot G + sk_D \cdot PK_i$ $\xrightarrow{C_i,\ i=1,\ldots,n}$ $C_i = f(i) \cdot G + sk_i \cdot PK_D$ $\xrightarrow{sk_i}$ $f(i) \cdot G$

Proof: $(d_1, \ldots, d_n) = H((C_i), (PK_i))$ publicly computable dual codeword (H random oracle)

$$V = \sum d_i C_i = \cancel{\sum d_i f(i) \cdot G}^{0} + \sum d_i sk_D \cdot PK_i = sk_D \sum d_i \cdot PK_i = sk_D \cdot U$$

π PoK: $sk_D = DL_G(PK_D) = DL_U(V)$ $\longrightarrow$ 2 Z_p elements

Remarks

Remarks

- See paper for proofs of correct resharing

Remarks

- See paper for proofs of correct resharing
- Parties can only reconstruct the group element $s \cdot G$, not exponent s

Remarks

- See paper for proofs of correct resharing
- Parties can only reconstruct the group element s·G, not exponent s
- Possible alternative using *Castagnos-Leguillaumie 15* (not in this paper).
  - “hard DDH groups with easy DL subgroups”.
  - Allow to reconstruct exponent.
  - Attainable with class groups.

Conclusions / Future Work

- We propose efficient encryption and PVSS for current winner, in the context of YOSO model

Conclusions / Future Work

- We propose efficient encryption and PVSS for current winner, in the context of YOSO model
- Our new PVSS is the most communication-efficient to date, has other applications such as random beacons

Conclusions / Future Work

- We propose efficient encryption and PVSS for current winner, in the context of YOSO model
- Our new PVSS is the most communication-efficient to date, has other applications such as random beacons
- Future work: PVSS under other assumptions
  - MPC would require to retrieve elements in $\mathbf{Z}_{p,}$ not in $\mathcal{G}$.
  - CL15, Paillier...

Thank you!

https://eprint.iacr.org/2022/242