Functional Encryption with Bounded Collusions via Multi-Party Computation

Sergey Gorbunov -- {U of Toronto}

Vinod Vaikuntanathan -- {U of Toronto}

Hoeteck Wee -- {George Washington U}

Public Key Encryption

Only Bob can decrypt and compute on m!

Bob

SK

Alice

PK

$CT = Enc(PK, m)$

Charlie

Public Key Encryption

How can we:

- **Allow Charlie to learn a function C of m?**
- **ensure Charlie doesn't learn more than C(m)?**
- **without asking Bob to do the work (outsourcing)**
- **and without asking Bob to be online (availability)**

Bob

SK

Alice

PK

$CT = Enc(PK, m)$

Charlie

Public Key Encryption

How can we:

- Allow Charlie to learn a function C of m?
- ~~ensure Charlie doesn't learn more than C(m)?~~
- without asking Bob to do the work (outsourcing)
- and without asking Bob to be online (availability)

Bob

SK

SK

Alice

PK

$CT = Enc(PK, m)$

Charlie

Public Key Encryption

How can we:

- Allow Charlie to learn a function C of m?
- ensure Charlie doesn't learn more than C(m)?
- ~~without asking Bob to do the work   (outsourcing)~~
- ~~and without asking Bob to be online (availability)~~

Bob

SK

$C(m)$

$CT = Enc(PK, m)$

Alice

PK

$CT = Enc(PK, m)$

Charlie

Fully Homomorphic Encryption [Gentry 09]

How can we:

- Allow Charlie to learn a function C of m?
- ensure Charlie doesn't learn more than C(m)?
- without asking Bob to do the work (outsourcing)
- ~~and without asking Bob to be online (availability)~~

Bob

SK

$Enc(C(m))$

$C(m)$

Alice

PK

$CT = Enc(PK, m)$

Charlie

$H.Eval(C, CT) = Enc(C(m))$

Functional Encryption [Boneh, Sahai, Waters 11] [O'Neill 10]

Allow Charlie to learn a function of M!

Bob

MSK

Alice

MPK

Charlie

C

Functional Encryption [Boneh, Sahai, Waters 11] [O'Neill 10]

Allow Charlie to learn a function of M!

Bob

MSK

*…Let **C** be a family of circuits and **M** be a message space*

Alice

MPK

Charlie

C

Functional Encryption [BSW'11, O'N10]

Allow Charlie to learn a function of M!

...*Let* ***C*** *be a family of circuits and* ***M*** *be a message space*

Bob

MSK

$SK = Keygen(MSK, C)$

(*Charlie no longer needs to communicate to Bob*)

Alice

MPK

Charlie

C

Functional Encryption [BSW'11, O'N10]

Allow Charlie to learn a function of M!

$\ldots$ *Let* $\boldsymbol{C}$ *be a family of circuits and* $\boldsymbol{M}$ *be a message space*

Bob

MSK

$SK = Keygen(MSK, C)$

Alice

MPK

$CT = Enc(MPK, m)$

Charlie

C

$Dec(SK, CT) = C(m)$

Functional Encryption [BSW'11, O'N10]

…Let $\boldsymbol{C}$ be a family of circuits and $\boldsymbol{M}$ be a message space

Security:
Adv should not learn anything about m, except $C(m)$

Bob

MSK

$SK = Keygen(MSK, C)$

Alice

MPK

$CT = Enc(MPK, m)$

Charlie

C

$Dec(SK, CT) = C(m)$

MOTIVATION MONDAYS!!!

MSK

$SK = Keygen(MSK, C)$

$C = circuit\ opening\ urgent\ emails$

MPK

$CT = Enc(MPK, email)$

C

$Dec(SK, CT) = email$ if urgent

$\perp$ otherwise

Special Cases of FE

- Identity-Based Encryption [Sha84, BF01, Coc01, BW06]

$$C_{id}(id', \mu) = \begin{cases} \mu \text{ if } id = id' \\ \perp \text{ otherwise} \end{cases}$$

- Fuzzy IBE [SW05]
- Attribute-Based Encryption [GPSW06, LOSTW10]
- Inner Product Predicate Encryption [KSW08, LOSTW10]

Can we construct functional encryption for all circuits?

$Can\ we\ construct\ functional\ encryption$
$for\ all\ circuits$?

$Yes\ we\ can$!

$with\ a\ small\ catch\ \ldots$

Functional Encryption

Allow Charlie to learn q functions of M (q is fixed before setup)

Security against $\boldsymbol{q-Bounded\ Collusions}$:
Adv should not learn anything about m, *except* $C_1(m), \ldots, C_q(m)$

Bob

MSK

$SK_1, \ldots, SK_q$

$SK_i = Keygen(MSK, C_i)$

Alice

MPK

$CT = Enc(MPK, m)$

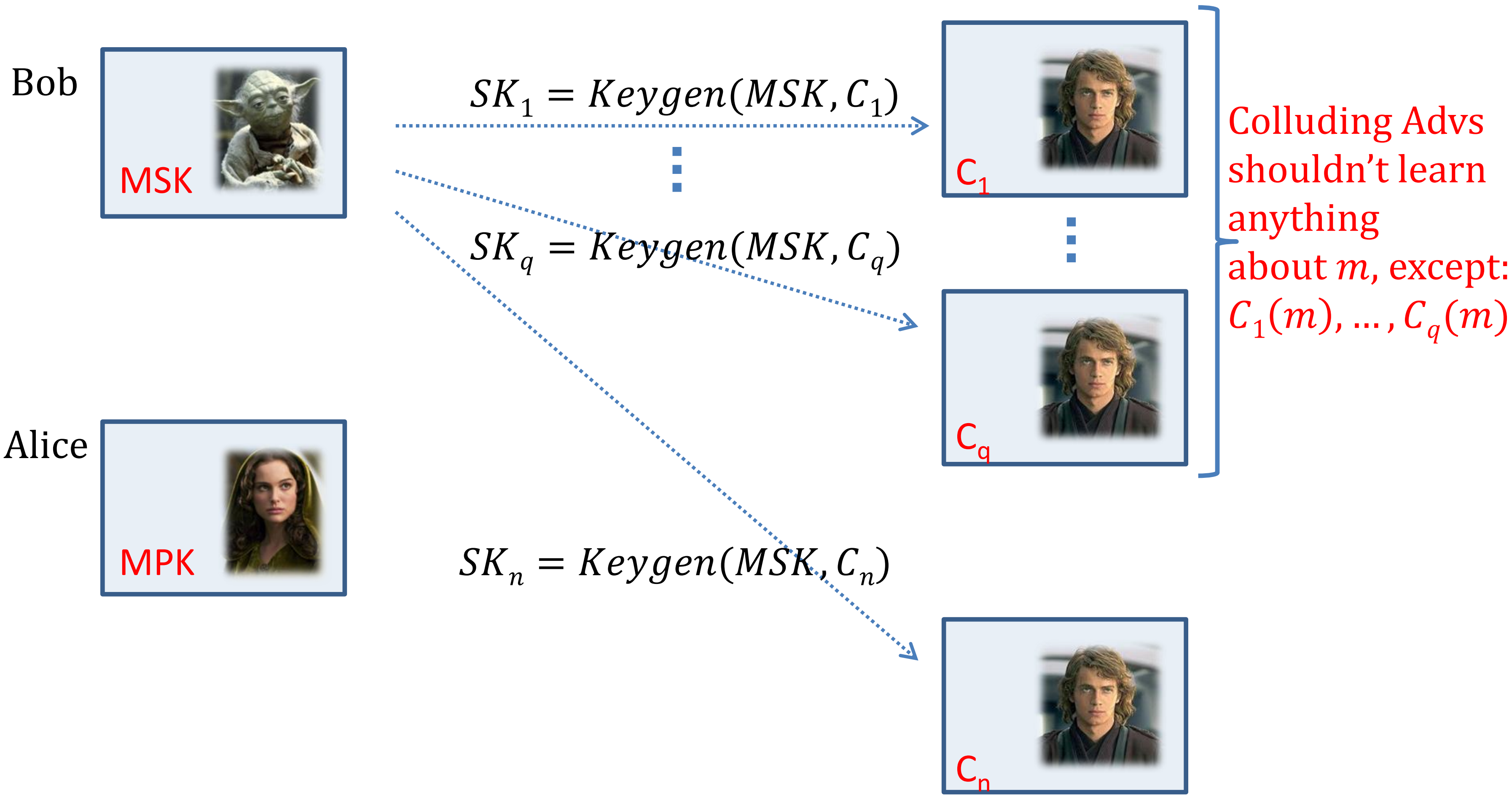
Charlie

C

$Dec(SK_1, CT) = C_1(m), \ldots,$
$Dec(SK_q, CT) = C_q(m)$

Functional Encryption

q-collusion security



Previous Work

q-collusion security

- Key-insulated public key cryptosystems
  [Dodis, Katz, Xu, Yung 02]
- Bounded CCA2
  [Cramer, Hanaoka, Hofheinz, Imai, Kiltz, Pass, Shelat, Vaikuntanathan 07]
- Bounded-collusion IBE
  [Goldwasser, Lewko, Wilson 12]

Our Result

Theorem: There exists a q-bounded non-adaptive simulation-secure FE scheme for all poly-size circuits, assuming:

- CPA-secure Public-key Encryption and
- PRGs computable in low-depth

Our Result

- Extends to adaptive for bounded # of messages

Theorem: There exists a q-bounded non-adaptive simulation-secure FE scheme for all poly-size circuits, assuming:

- CPA-secure Public-key Encryption and
- PRGs computable in low-depth

Our Result

Theorem: There exists a q-bounded non-adaptive simulation-secure FE scheme for all poly-size circuits, assuming:

- CPA-secure Public-key Encryption and
- PRGs computable in low-depth

- factoring
- discrete logarithm
- lattice problems

Our Result

Theorem: There exists a q-bounded non-adaptive simulation-secure **public index predicate encryption** scheme for all poly-size circuits, assuming:

- CPA-secure Public-key Encryption ~~and~~
- ~~PRGs computable in low depth~~

Our Result

Theorem: There exists a q-bounded non-adaptive simulation-secure FE scheme for all poly-size circuits, assuming:

- CPA-secure Public-key Encryption and
- PRGs computable in low-depth

Remark 1:

[Thm: Agrawal, **G**, Vaikuntanathan, Wee 12]

*For **unbounded** collusions, it is **impossible** to achieve non – adaptive simulation secure FE for all circuits.*

Our Result

Theorem: There exists a q-bounded non-adaptive simulation-secure FE scheme for all poly-size circuits, assuming:

- CPA-secure Public-key Encryption and
- PRGs computable in low-depth

Remark 2:

[Thm: Boneh, Sahai, Waters 11]

*It is **impossible** to achieve **adaptive** simulation secure FE for all circuits.*

(*many messages*, 1 *SK*)

Our Result

Theorem: There exists a q-bounded non-adaptive simulation-secure FE scheme for all poly-size circuits, assuming:

- CPA-secure Public-key Encryption and
- PRGs computable in low-depth

Remark 3:

$$\mathit{Simulation\ Security} \rightarrow \mathit{IND\ security}$$

Roadmap

1-FE for arbitrary circuits [Sahai, Seyalioglu 10]

↓ Using MPC [Ben-Or, Goldwasser, Wigderson 88]

q-FE for degree-d circuits

↓ FE Bootstrapping Theorem: Using Randomized Encodings [Applebaum, Ishai, Kushilevitz 05] [Yao 86]

q-FE for arbitrary circuits

Roadmap

1-FE for arbitrary circuits [Sahai, Seyalioglu 10]

Using MPC
[Ben-Or, Goldwasser, Wigderson 88]

q-FE for degree-d circuits

Class of functions:

- Computes bounded degree polynomial
- $for\ all\ C \in \boldsymbol{C},$
  $C(\cdot)\ is\ l - variate\ polynomial\ over\ \mathbb{F}\ of\ degree\ d$

1-FE for all circuits

[Sahai, Seyalioglu 10]

Ciphertext CT: A universal garbled circuit encoding m [Yao 82]

Secret key SK^C: Set of input labels

It is correct but **NOT** secure for two sets of input labels! (i.e. **insecure for q=2**)

q-bounded Collusions FE

C(•) is a degree d polynomial

Shamir's SS [Shamir 79]

Important property: Given two shares $s_1(i)$ and $s_2(i)$, we can perform computation over the shares! [Ben-Or, Goldwasser, Wigderson 88]

$s_1(i) + s_2(i) = (s_1 + s_2)(i)$ (additive homomorphism)

$s_1(i) * s_2(i) = (s_1 * s_2)(i)$ (multiplicative homomorphism)

q-bounded Collusions FE

C(•) is a degree d polynomial

Shamir's SS [Shamir 79]

Important property: Given two shares $s_1(i)$ and $s_2(i)$, we can perform computation over the shares! [Ben-Or, Goldwasser, Wigderson 88]

$s_1(i) + s_2(i) = (s_1 + s_2)(i)$ (additive homomorphism)

$s_1(i) * s_2(i) = (s_1 * s_2)(i)$ (multiplicative homomorphism)

Catch:
Degree of the underlying polynomial increases with each multiplication!

q-bounded Collusions FE

C(•) is a degree d polynomial

$\boldsymbol{Parameters}: \boldsymbol{N} = \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{q}), \boldsymbol{t} = \boldsymbol{t}(\boldsymbol{q}),$ 1-FE: ($Setup^1$, $Keygen^1$, Enc^1, Dec^1)

Setup: Run $Setup^1$ N times:

MPK_1	MPK_2	MPK_3	…	MPK_{N-1}	MPK_N
MSK_1	MSK_2	MSK_3	…	MSK_{N-1}	MSK_N

q-bounded Collusions FE

C(•) is a degree d polynomial

$\boldsymbol{Parameters}: \boldsymbol{N} = \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{q}), \boldsymbol{t} = \boldsymbol{t}(\boldsymbol{q}),$ 1-FE: ($Setup^1$, **$Keygen^1$**, Enc^1, Dec^1)

Setup: Run $Setup^1$ N times:

MPK_1 MPK_2 MPK_3 … MPK_{N-1} MPK_N

MSK_1 MSK_2 MSK_3 … MSK_{N-1} MSK_N

$Keygen^1_{MSK(1)}(C)$ $Keygen^1_{MSK(2)}(C)$ $Keygen^1_{MSK(N-1)}(C)$

$Keygen_{MSK}(C)$: SK_1^C SK_2^C … SK_{N-1}^C

Random subset S of secret keys $\{MSK_i\}$ is chosen
Run $Keygen^1$ on C for all MSK_i in S

q-bounded Collusions FE

C(•) is a degree d polynomial

$\boldsymbol{Parameters}: \boldsymbol{N = N(d,q), t = t(q)}$, 1-FE: ($Setup^1$, $Keygen^1$, **Enc^1**, Dec^1)

Setup: Run $Setup^1$ N times:

	MPK_1	MPK_2	MPK_3	…	MPK_{N-1}	MPK_N
	MSK_1	MSK_2	MSK_3	…	MSK_{N-1}	MSK_N
$Keygen_{MSK}(C)$:	SK_1^C	SK_2^C		…	SK_{N-1}^C	
$Enc_{MPK}(m)$:	m_1	m_2	m_3	…	m_{N-1}	m_N
	MPK_1	MPK_2	MPK_3		MPK_{N-1}	MPK_N
	CT_1	CT_2	CT_3	…	CT_{N-1}	CT_N

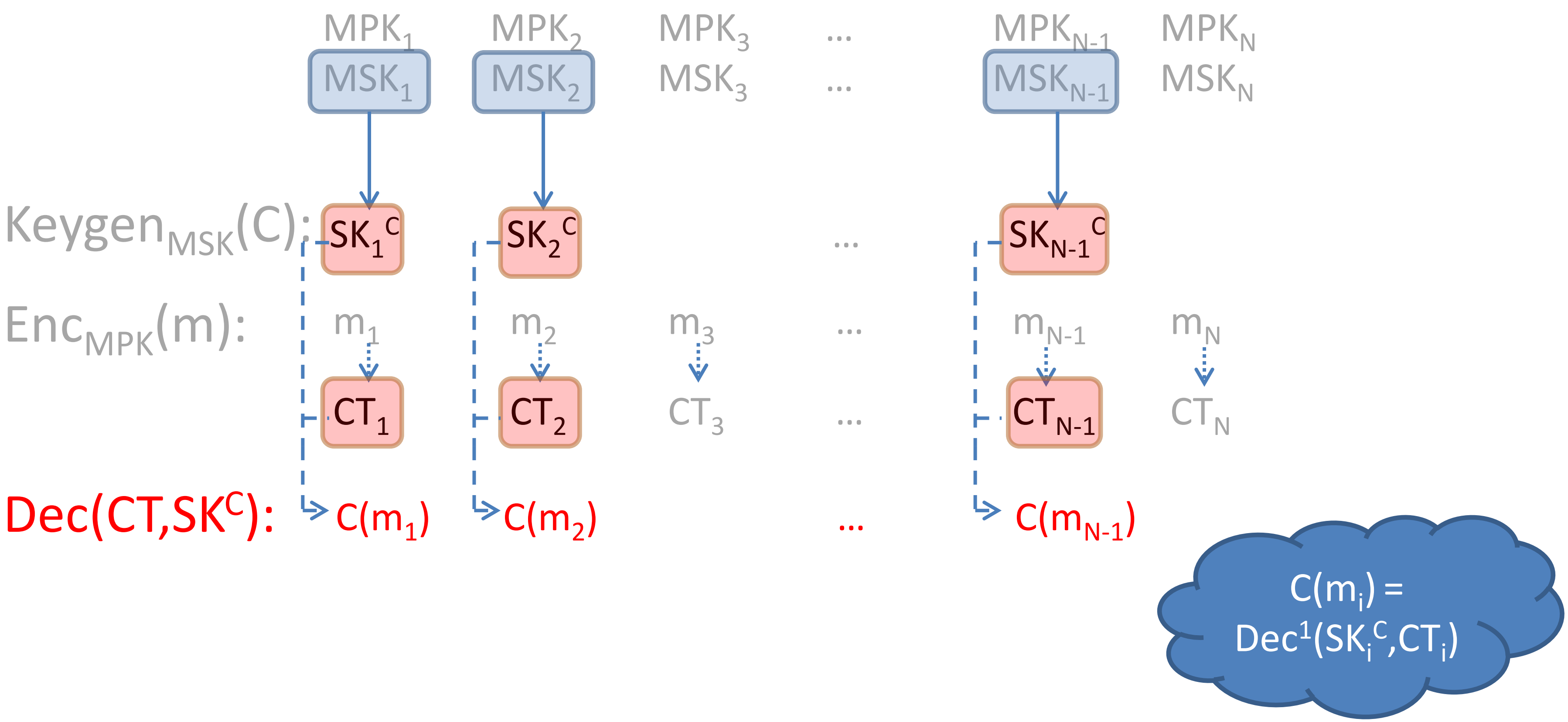
Share m -> $(m_1,\ldots,m_N)$ using degree t polynomial

q-bounded Collusions FE

C(•) is a degree d polynomial

$\boldsymbol{Parameters}: \boldsymbol{N} = \boldsymbol{N}(\boldsymbol{d}, \boldsymbol{q}), \boldsymbol{t} = \boldsymbol{t}(\boldsymbol{q}),$ 1-FE: ($Setup^1$, $Keygen^1$, Enc^1, **Dec^1**)

Setup: Run $Setup^1$ N times:



q-bounded Collusions FE

C(•) is a degree d polynomial

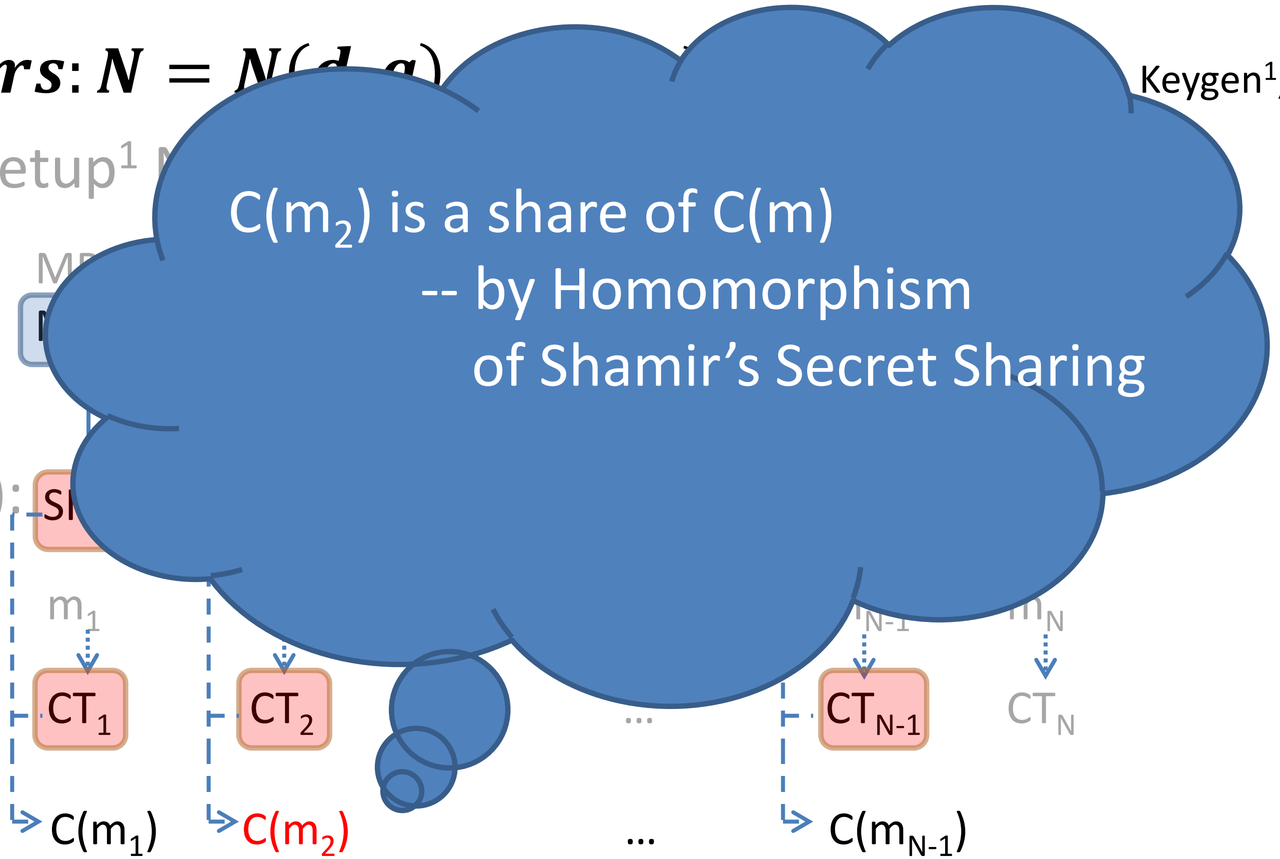
$\boldsymbol{Parameters}: \boldsymbol{N = N(d, q)}$ ... Keygen[1], Enc[1], **Dec[1]**)

Setup: Run Setup[1] ...

Keygen$_{MSK}$(C):

Enc$_{MPK}$(m):

Dec(CT,SK^C):

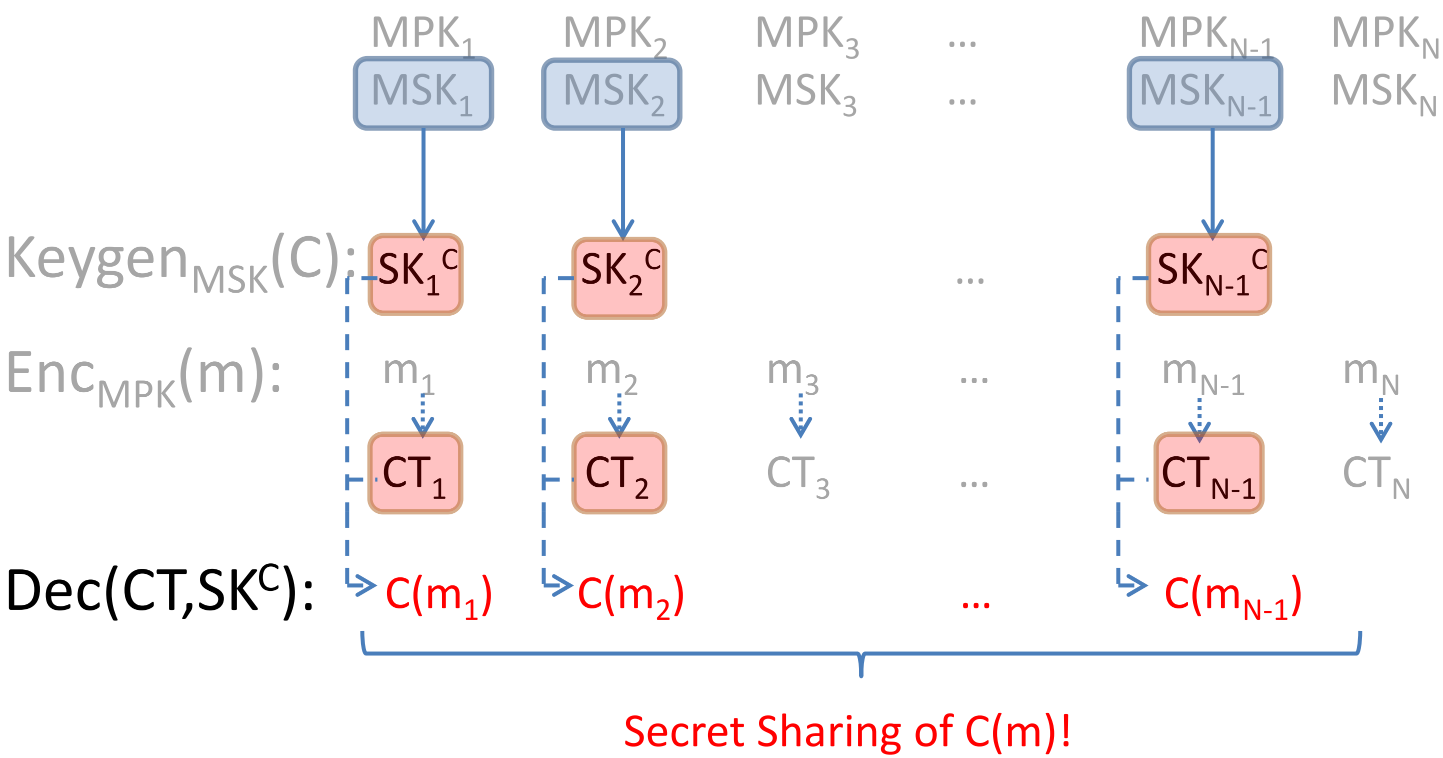


q-bounded Collusions FE

C($\bullet$) is a degree d polynomial

$\boldsymbol{Parameters}: \boldsymbol{N = N(d,q), t = t(q)}$, 1-FE: ($Setup^1$, $Keygen^1$, Enc^1, **Dec^1**)

Setup: Run $Setup^1$ N times:

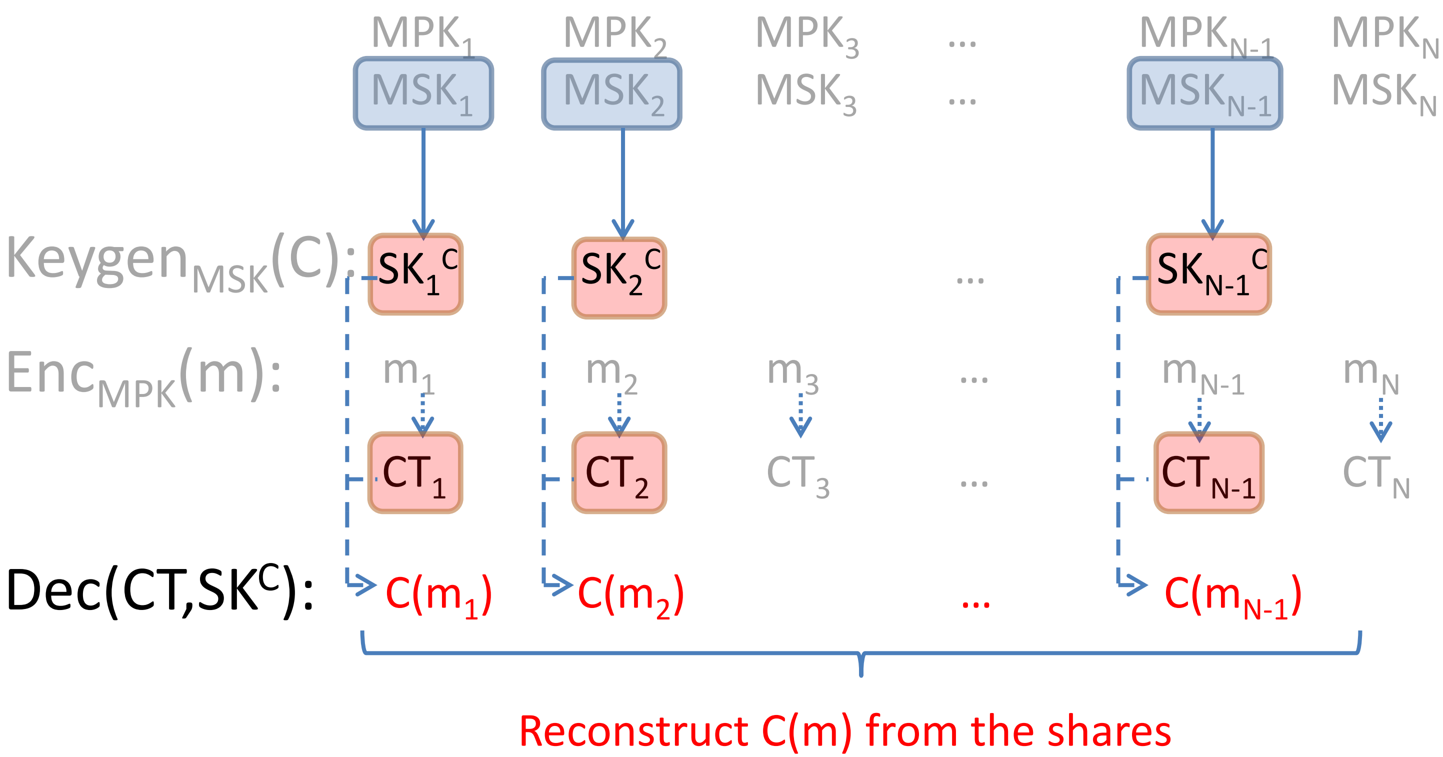


q-bounded Collusions FE

C(•) is a degree d polynomial

$\boldsymbol{Parameters}: \boldsymbol{N = N(d,q), t = t(q)},$ 1-FE: (Setup[1], Keygen[1], Enc[1], **Dec[1]**)

Setup: Run Setup[1] N times:



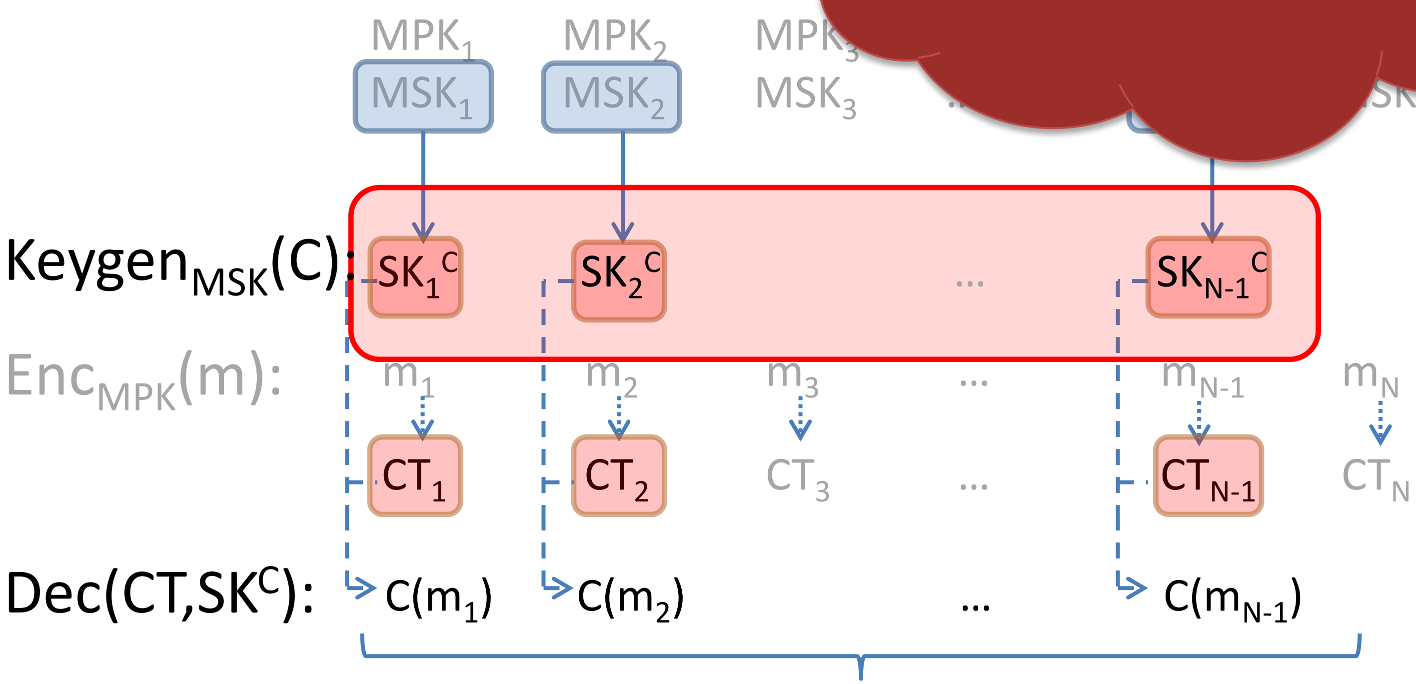
q-bounded Collusions FE

C(•) is a degree d polynomial

Parameters: $N = N(d,q)$, ... c[1], Dec[1])

Setup: Run Setup[1] N times:

Correctness:
$m_i = s(i)$, where s() is degree t,
C(•) is degree d
→ C(s(i)) is degree dt polynomial
→ Give dt+1 SK's



Reconstruct C(m) from the shares

q-bounded Collusions FE

C(•) is a degree d polynomial

$\boldsymbol{Parameters}: \boldsymbol{N = N(d, q), t = t(q)}$, 1-FE: ($Setup^1$, $Keygen^1$, Enc^1, **Dec^1**)

Setup: Run $Setup^1$ N times:

MPK_1 MPK_2 MPK_3 … MPK_{N-1} MPK_N

MSK_1 MSK_2 MSK_3 … MSK_{N-1} MSK_N

$Keygen_{MSK}(C_1)$: SK_1^{C1} SK_2^{C1} … SK_{N-1}^{C1}

$Keygen_{MSK}(C_2)$: SK_2^{C2} SK_3^{C2} … SK_N^{C2}

$Enc_{MPK}(m)$: m_1 m_2 m_3 … m_{N-1} m_N

CT_1 CT_2 CT_3 … CT_{N-1} CT_N

$Dec(CT, SK^C)$:

q-bounded Collusions FE

C(•) is a degree d polynomial

$\boldsymbol{Parameters}: \boldsymbol{N = N(d,q), t = t(q)},$ 1-FE: ($Setup^1$, $Keygen^1$, Enc^1, **Dec^1**)

Setup: Run $Setup^1$ N times:

	MPK_1	MPK_2	MPK_3	…	MPK_{N-1}	MPK_N
	MSK_1	MSK_2	MSK_3	…	MSK_{N-1}	MSK_N
$Keygen_{MSK}(C_1)$:	SK_1^{C1}	SK_2^{C1}		…	SK_{N-1}^{C1}	
$Keygen_{MSK}(C_2)$:		SK_2^{C2}	SK_3^{C2}	…		SK_N^{C2}
$Enc_{MPK}(m)$:	m_1	m_2	m_3	…	m_{N-1}	m_N
	CT_1	CT_2	CT_3	…	CT_{N-1}	CT_N
$Dec(CT,SK^C)$:	$C_1(m_1)$	$C_1(m_2)$		…	$C_1(m_{N-1})$	
		$C_2(m_2)$	$C_2(m_3)$	…		$C_2(m_{N-1})$

q-bounded Collusions FE

C(•) is a degree d polynomial

$Parameters: N = N(d,q), t = t(q),$ 1-FE:

Security (intuition):
We are OK, given that the Decryptor learns <= t shares

Setup: Run Setup[1] N times:

MPK_1 MPK_2 MPK_3 …

MSK_1 MSK_2 MSK_3 … MSK_N

$Keygen_{MSK}(C_1)$: SK_1^{C1} SK_2^{C1} … SK_{N-1}^{C1}

$Keygen_{MSK}(C_2)$: SK_2^{C2} SK_3^{C2} … SK_N^{C2}

$Enc_{MPK}(m)$: m_1 m_2 m_3 … m_{N-1} m_N

CT_1 CT_2 CT_3 … CT_{N-1} CT_N

$Dec(CT,SK^C)$: $C_1(m_1)$ $C_1(m_2)$ … $C_1(m_{N-1})$

$C_2(m_2)$ $C_2(m_3)$ … $C_2(m_{N-1})$

+ m_2

q-bounded Collusions FE

C(•) is a degree d polynomial

Technical Problem 1:

- Adversary learns shares $C(m_i)$, so the simulator must be able to simulate them. However, these are not random shares, so unclear how to simulate. (known problem in BGW)

q-bounded Collusions FE

C(•) is a degree d polynomial

Technical Problem 1:

- Adversary learns shares $C(m_i)$, so the simulator must be able to simulate them. However, these are not random shares, so unclear how to simulate. (known problem in BGW)

Solution

- Randomize each share $C(m_i)$ by adding random share r_i of 0
  $C'(m_i||r_i) = C(m_i) + r_i$

q-bounded Collusions FE

C(•) is a degree d polynomial

Technical Problem 1:

- Adversary learns shares $C(m_i)$, so the simulator must be able to simulate them. However, these are not random shares, so unclear how to simulate. (known problem in BGW)

Solution

- Randomize each share $C(m_i)$ by adding random share r_i of 0
  $C'(m_i||r_i) = C(m_i) + r_i$

Technical Problem 2:

- Adding random shares of 0 of the same polynomial creates correlation between shares of $C_1(m) \ldots C_q(m)$

q-bounded Collusions FE

C(•) is a degree d polynomial

Technical Problem 1:

- Adversary learns shares $C(m_i)$, so the simulator must be able to simulate them. However, these are not random shares, so unclear how to simulate. (known problem in BGW)

Solution

- Randomize each share $C(m_i)$ by adding random share r_i of 0
  $C'(m_i||r_i) = C(m_i) + r_i$

Technical Problem 2:

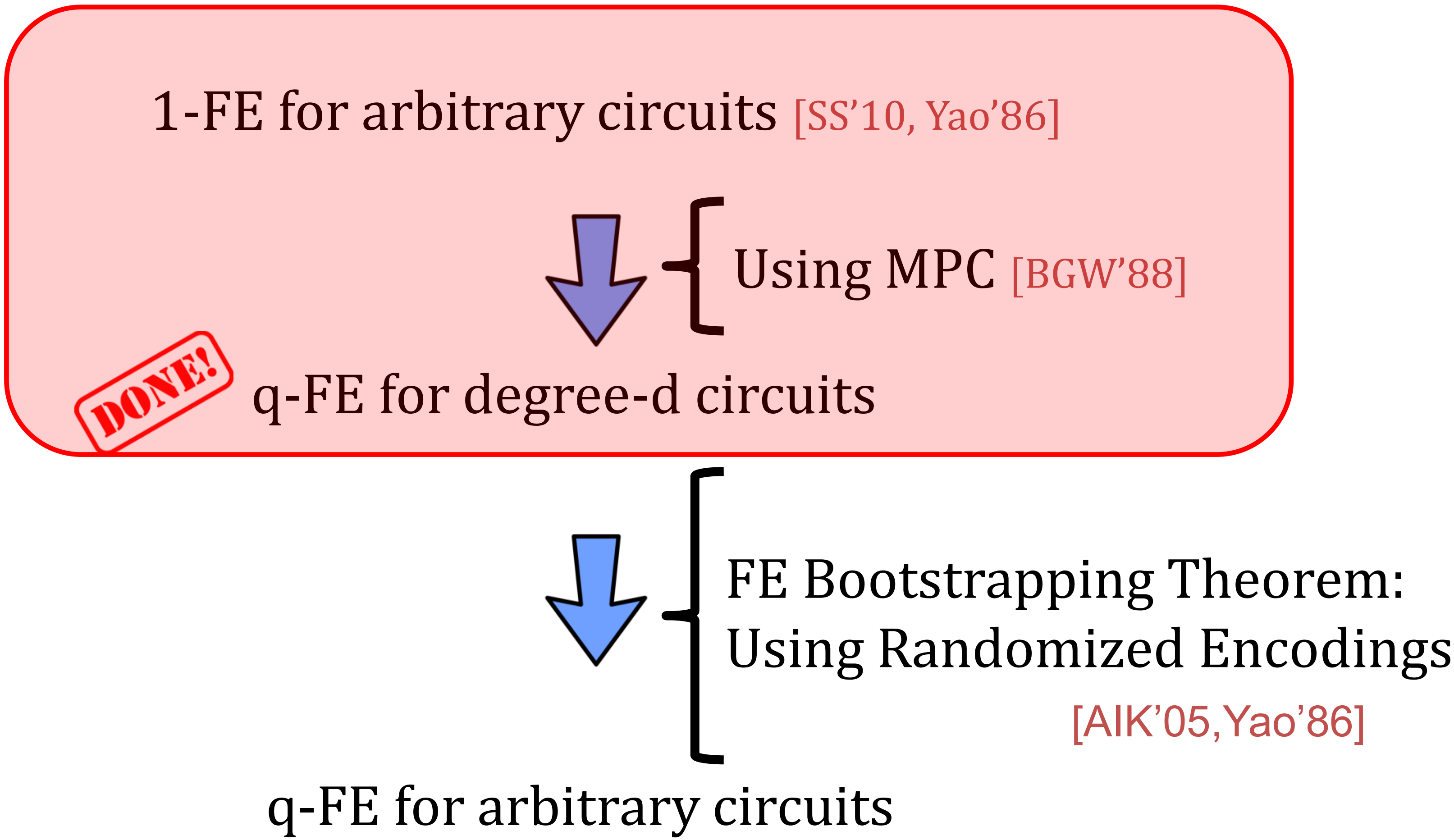
- Adding random shares of 0 of the same polynomial creates correlation between shares of $C_1(m)$ … $C_q(m)$

Solution

- Add a **q**-wise independent random shares of 0

$$C'_w(m||\vec{r_i}) = C(m_i) + \sum_{j \in w} r_i[j]$$

q-bounded Collusions FE



q-bounded Collusions FE

q-FE for degree-d circuits

FE Bootstrapping Theorem:
Using Randomized Encodings
[Applebaum, Ishai, Kushilevitz 05]
[Yao 86]

q-FE for arbitrary circuits

Idea: A function computing a randomized encoding for C is of low degree. (assuming low degree PRG) [AIK05]

q-bounded Collusions FE

1-FE for arbitrary circuits [SS'10, Yao'86]

↓ Using MPC [BGW'88]

q-FE for degree-d circuits

↓ FE Bootstrapping Theorem: Using Randomized Encodings [AIK'05,Yao'86]

q-FE for arbitrary circuits

q-bounded Collusions FE

1-FE for arbitrary circuits [SS'10, Yao'86]

Using MPC [BGW'88]

q-FE for degree-d circuits

FE Bootstrapping Theorem:
Using Randomized Encodings

[AIK'05,Yao'86]

q-FE for arbitrary circuits

Open Problems:

- IND-secure FE for all circuits (unbounded collusions)?
- New connections amongst MPC, ZK and FE?


Thank you!

Back – up slide 1

Small Pairwise Intersection:

Let $S_1, S_2, \ldots, S_n \in [N]$. Want to make sure:

$$|\cup_{i \neq j} (S_i \cap S_j)| \leq t$$

Cover-Freeness:

Let $w_1, w_2, \ldots, w_n \in [N]$. Want to make sure:

$$\text{For all } i \in [q],\ w_i \setminus (\cup_{i \neq j} w_j) \neq \emptyset$$

Back – up slide 2

Class of functions:

- Deterministic

- Computes bounded degree polynomial

- $M = \mathbb{F}^l, for\ all\ C,$
  $C(\cdot)\ is\ l - variate\ polynomial\ over\ \mathbb{F}\ of\ degree\ d$

- Handles arithmetic and boolean circuits (Set $\mathbb{F}$ to be a large extension of $\mathbb{F}_2$) (constant fan-in)