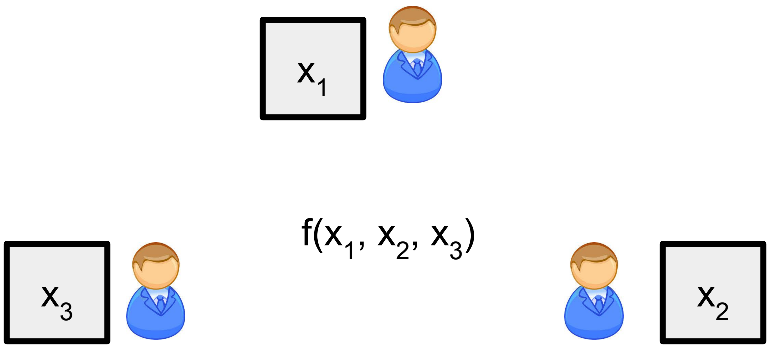
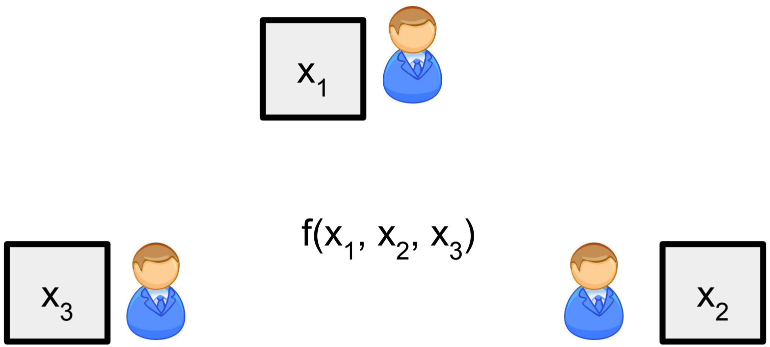
Better 2-round adaptive MPC

Ran Canetti, Oxana Poburinnaya, Muthuramakrishnan Venkitasubramaniam

Secure Multiparty Computation[Yao'82]



Secure Multiparty Computation[Yao'82]



Correctness: every party learns $y = f(x_1, x_2, x_3)$
Security: even if a party is dishonest, it only learns the output y, but nothing else

Our results :

Semi-honest case

2 round fully adaptive MPC with useful properties (randomness-hiding, RAM-efficient, global CRS...)

Our results :

Semi-honest case

2 round fully adaptive MPC with useful properties (randomness-hiding, RAM-efficient, global CRS...)

malicious case

GP’15 2 round fully adaptive MPC becomes **RAM-efficient**

Our results :

Semi-honest case

2 round fully adaptive MPC with useful properties (randomness-hiding, RAM-efficient, global CRS...)

malicious case

ZK proofs with RAM efficiency

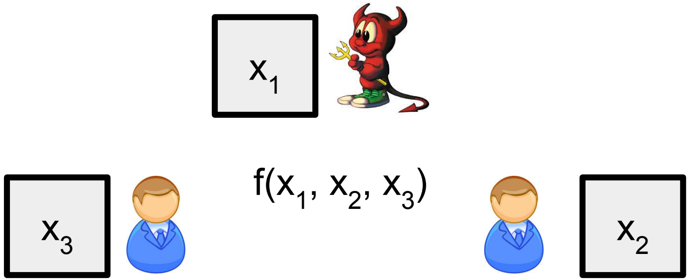
plug into GP’15

GP’15 2 round fully adaptive MPC becomes **RAM-efficient**

Static vs Adaptive Security

when do parties become dishonest?

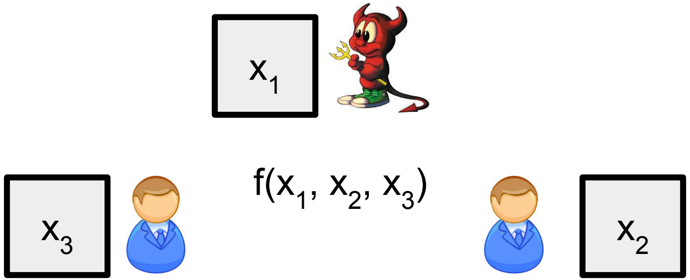
Static security:
a set of dishonest parties is fixed
before the protocol starts



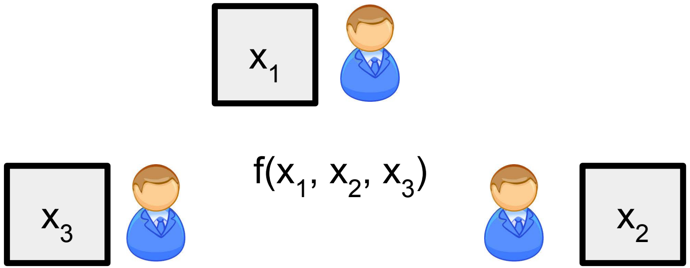
Static vs Adaptive Security

when do parties become dishonest?

Static security:
a set of dishonest parties is fixed
before the protocol starts



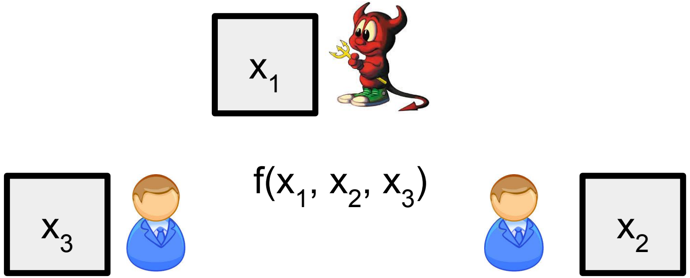
Adaptive security:
parties may become dishonest
during the execution of the protocol



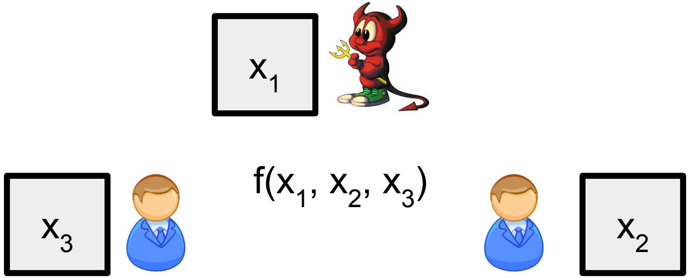
Static vs Adaptive Security

when do parties become dishonest?

Static security:
a set of dishonest parties is fixed
before the protocol starts



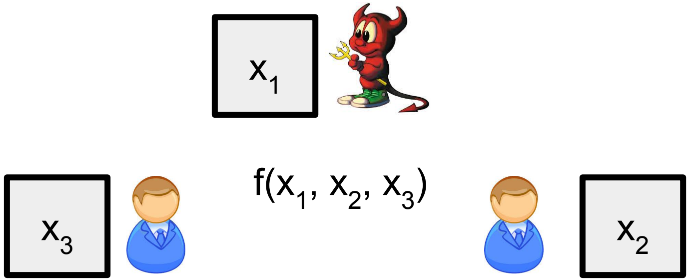
Adaptive security:
parties may become dishonest
during the execution of the protocol



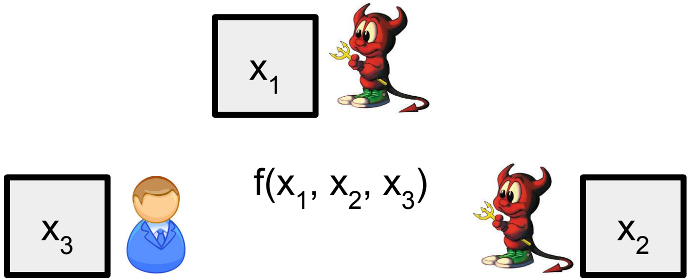
Static vs Adaptive Security

when do parties become dishonest?

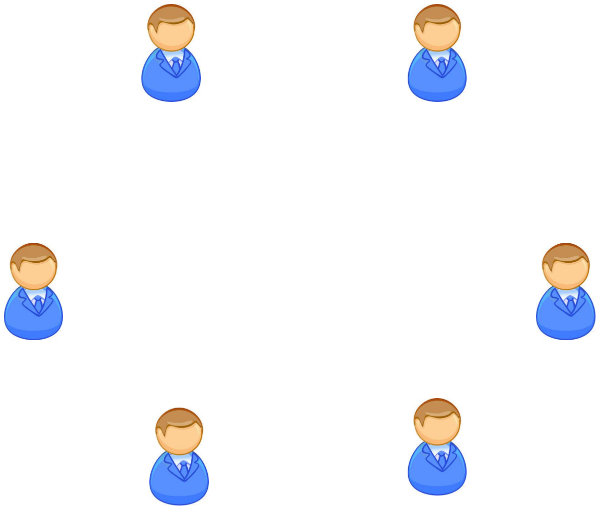
Static security:
a set of dishonest parties is fixed
before the protocol starts



Adaptive security:
parties may become dishonest
during the execution of the protocol

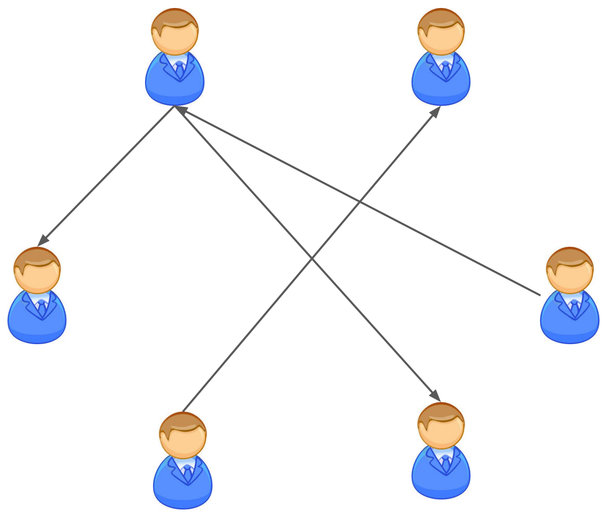


Adaptive Security of MPC



Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Adaptive Security of MPC

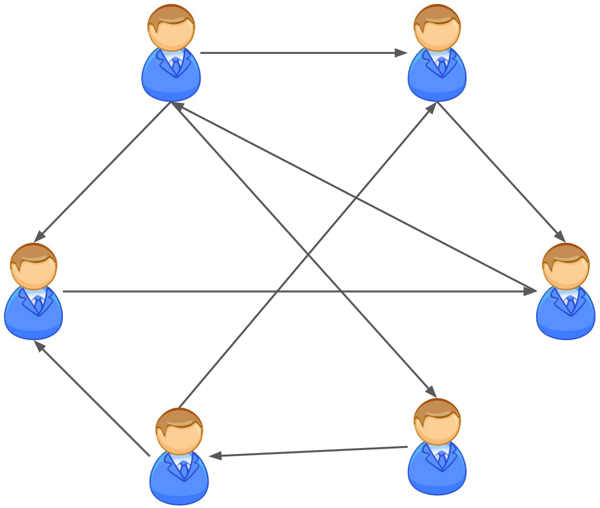


Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:

1. simulate communication (without knowing $x_1, \ldots, x_n$)

Adaptive Security of MPC

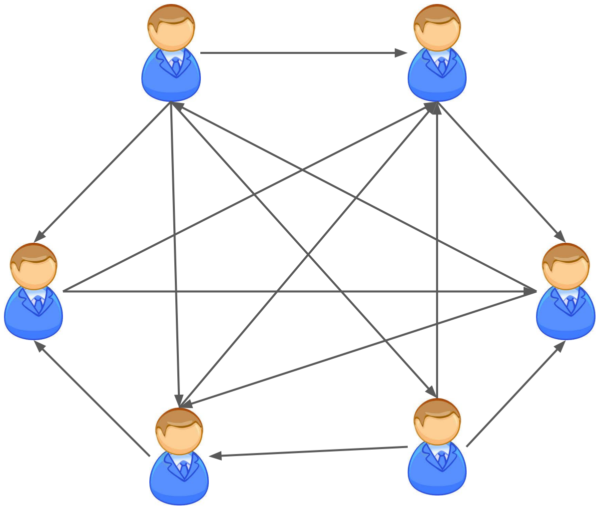


Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:

1. simulate communication (without knowing $x_1, \ldots, x_n$)

Adaptive Security of MPC

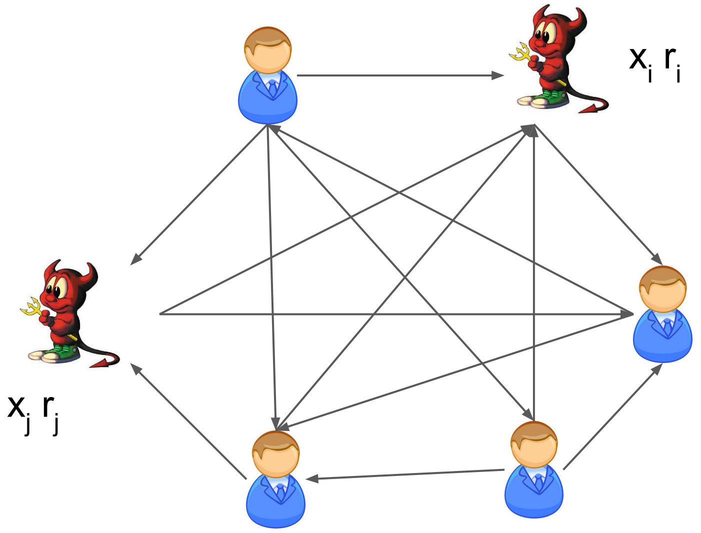


Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:

1. simulate communication (without knowing $x_1, \ldots, x_n$)

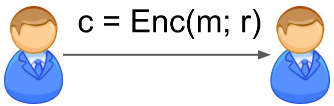
Adaptive Security of MPC



Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:
1. simulate communication (without knowing $x_1, \ldots, x_n$)
2. simulate r_i of corrupted parties, consistent with communication and x_i

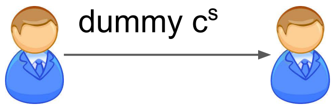
Example: Adaptively Secure Encryption (NCE)



Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:

Example: Adaptively Secure Encryption (NCE)

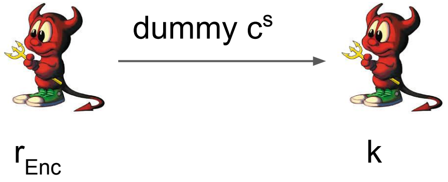


Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:

1. Sim() → c^s, state

Example: Adaptively Secure Encryption (NCE)

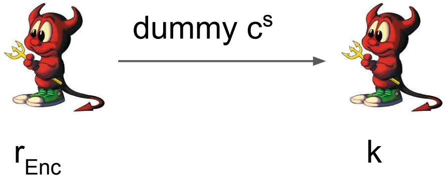


Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:

1. $Sim() \rightarrow c^s$, state
2. $Sim(state, m) \rightarrow r^s_{Enc}, k^s$

Example: Adaptively Secure Encryption (NCE)



Adaptive corruptions:
adversary can decide who to corrupt adaptively during the execution

Simulator:

1. $Sim() \rightarrow c^s$, state
2. $Sim(state, m) \rightarrow r^s_{Enc}, k^s$

Adv gets:

- either real $(r, k, c = Enc_k(m; r))$
- or fake (r^s_{Enc}, k^s, c^s)

possible for a non-committing encryption (NCE)

Full Adaptive Security

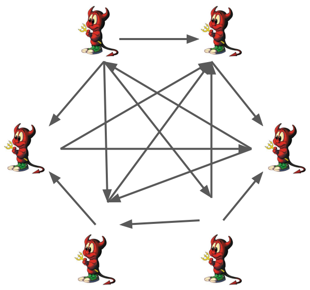
Full adaptive security:

- No erasures

Full Adaptive Security

Full adaptive security:

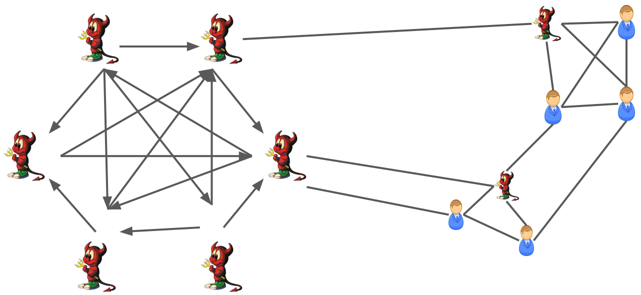
- No erasures
- Security even when all parties are corrupted



Full Adaptive Security

Full adaptive security:

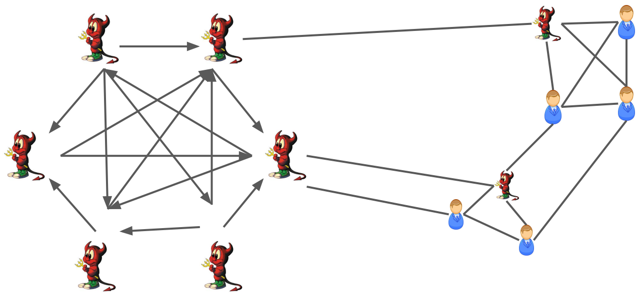
- No erasures
- Security even when all parties are corrupted



Full Adaptive Security

Full adaptive security:

- No erasures
- Security even when all parties are corrupted



Until 2015: # of rounds ~ depth of circuit (CLOS02)

Constant round protocols: CGP15, DKR15, GP15.

Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions
CGP'15	2	2	OWF subexp iO
DKR'15	n	4	OWF iO
GP'15	n	2	TDP subexp. iO

← the only 2 round MPC

*need a CRS even for HBC case!

Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions
CGP’15	2	2	OWF subexp iO
DKR’15	n	4	OWF iO
GP’15	n	2	TDP subexp. iO

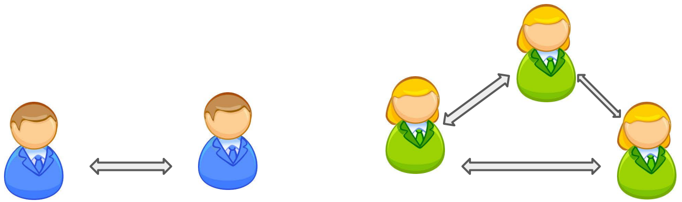
Q1: can we build 2 round MPC with **global (non-programmable)** CRS?

Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions	global CRS
CGP’15	2	2	OWF subexp iO	+
DKR’15	n	4	OWF iO	+
GP’15	n	2	TDP subexp. iO	- (even in HBC case)

Q1: can we build 2 round MPC with **global (non-programmable)** CRS?

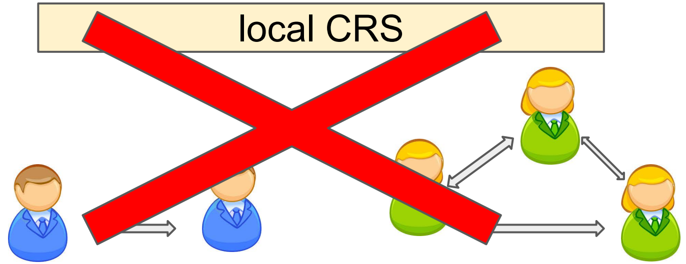
local CRS



Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions	global CRS
CGP'15	2	2	OWF subexp iO	+
DKR'15	n	4	OWF iO	+
GP'15	n	2	TDP subexp. iO	- (even in HBC case)

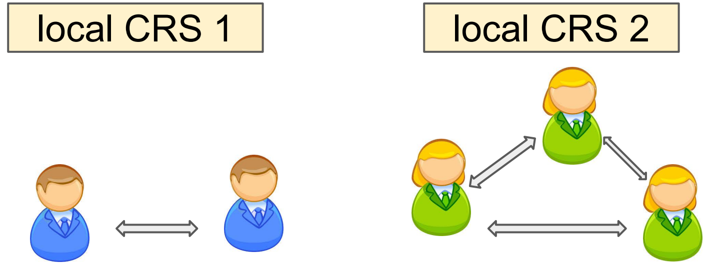
Q1: can we build 2 round MPC with **global (non-programmable)** CRS?



Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions	global CRS
CGP'15	2	2	OWF subexp iO	+
DKR'15	n	4	OWF iO	+
GP'15	n	2	TDP subexp. iO	- (even in HBC case)

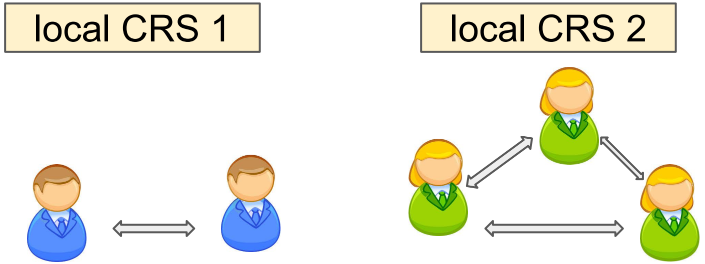
Q1: can we build 2 round MPC with **global (non-programmable)** CRS?

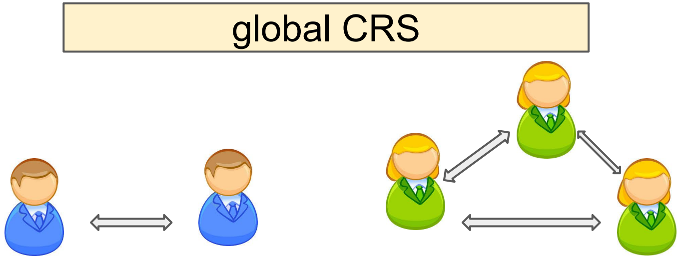


Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions	global CRS
CGP'15	2	2	OWF subexp iO	+
DKR'15	n	4	OWF iO	+
GP'15	n	2	TDP subexp. iO	- (even in HBC case)

Q1: can we build 2 round MPC with **global (non-programmable)** CRS?





Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions	global CRS	randomness hiding
CGP'15	2	2	OWF subexp iO	+	+
DKR'15	n	4	OWF iO	+	+
GP'15	n	2	TDP subexp. iO	- (even in HBC case)	-

Q2: can we achieve **randomness hiding?** (Evaluation of $f(x_1, \ldots, x_n; r)$ hides r even if everyone is corrupted)

choose N = pq

nobody knows p, q

Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions	global CRS	randomness hiding	supports RAM
CGP'15	2	2	OWF subexp iO	+	+	-
DKR'15	n	4	OWF iO	+	+	-
GP'15	n	2	TDP subexp. iO	- (even in HBC case)	-	-

Q3: can we use the fact that f is a succinct RAM program?

Full Adaptive Security: state of the art, semi-honest

	# of parties	# of rounds	assumptions	global CRS	randomness hiding	supports RAM
CGP'15	2	2	OWF subexp iO	+	+	-
DKR'15	n	4	OWF iO	+	+	-
GP'15	n	2	TDP subexp. iO	- (even in HBC case)	-	-

Q4: can we build 2 round MPC from **weaker assumptions?** (e.g. remove the need for subexp. iO)

Full Adaptive Security

	# of parties	# of rounds	assumptions	global CRS	randomness hiding	supports RAM
CGP'15	2	2	OWF subexp iO	+	+	-
DKR'15	n	4	OWF iO	+	+	-
GP'15	n	2	TDP subexp. iO	- (even in HBC case)	-	-
This work	**n**	**2**	**OWF iO**	**+**	**+**	**+**

Full Adaptive Security

	# of parties	# of rounds	assumptions	global CRS	randomness hiding	supports RAM
CGP’15	2	2	OWF subexp iO	+	+	-
DKR’15	n	4	OWF iO	+	+	-
GP’15	n	2	TDP subexp. iO	- (even in HBC case)	-	-
This work	**n**	**2**	**OWF** **iO**	**+**	**+**	**+**

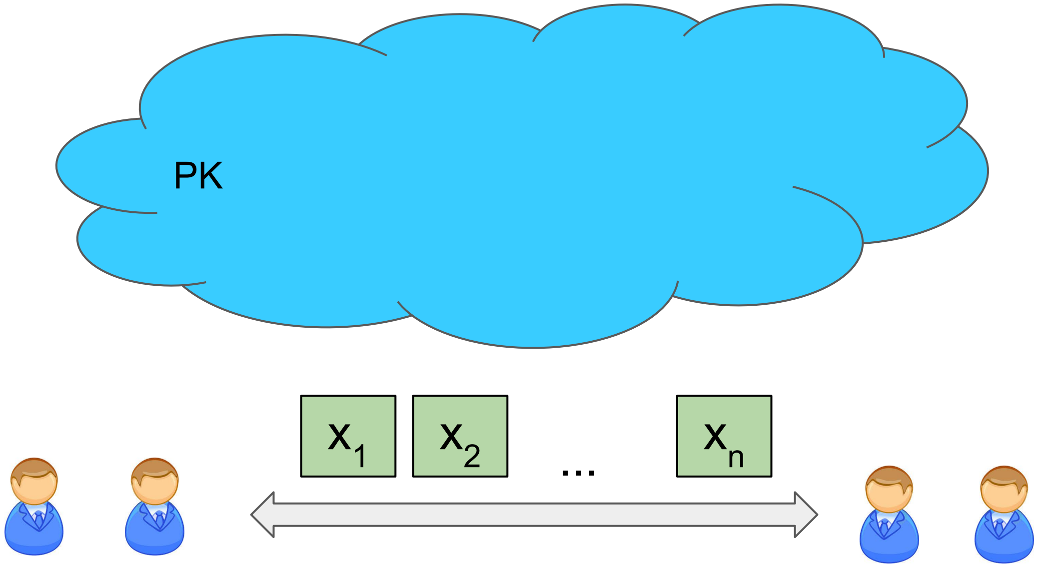
Subsequent work

HPV’16	2	2	hardware tokens OWF	no CRS	-	-
CPV’16	2 (n)	2 (const)	NCE*	no CRS	-	-

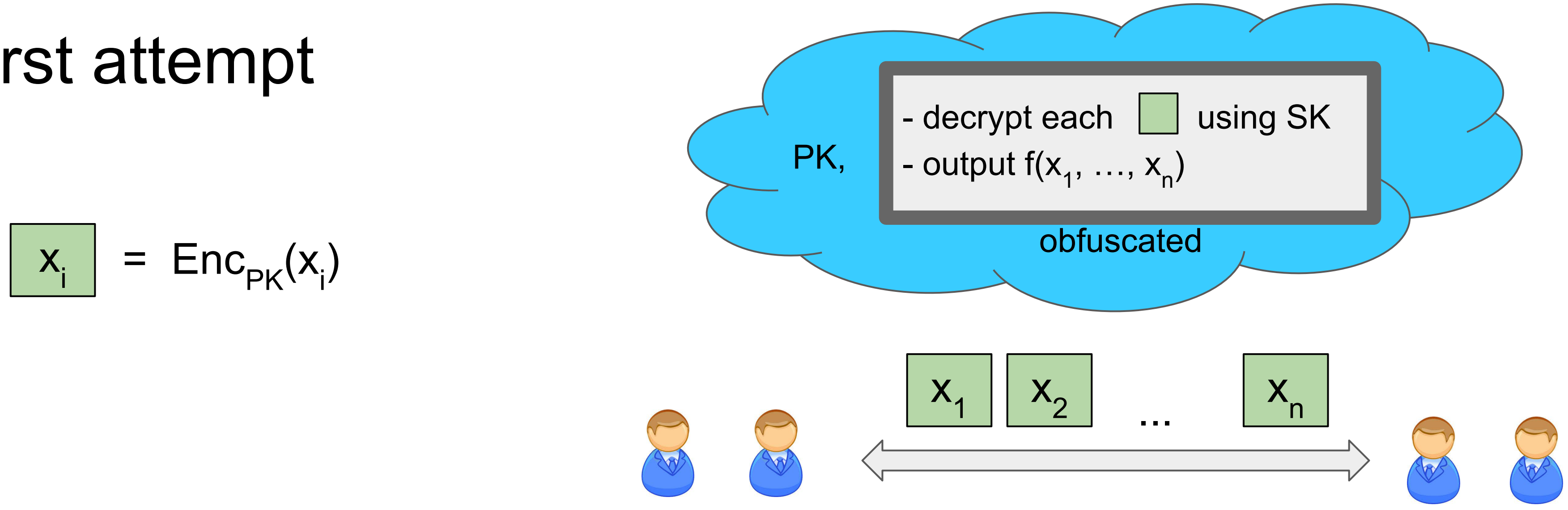
Part I: HBC protocol with global CRS

First attempt

$\boxed{x_i} = \mathrm{Enc}_{PK}(x_i)$

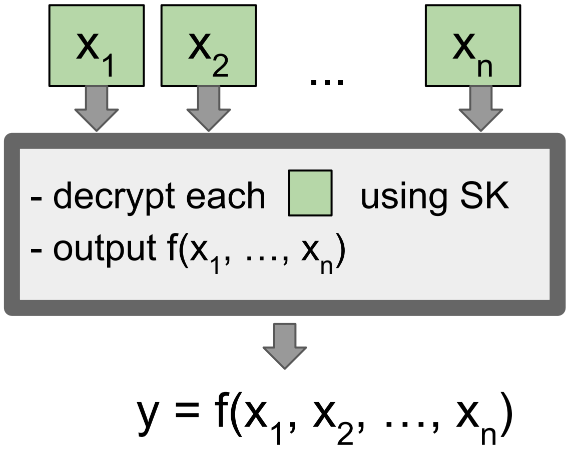


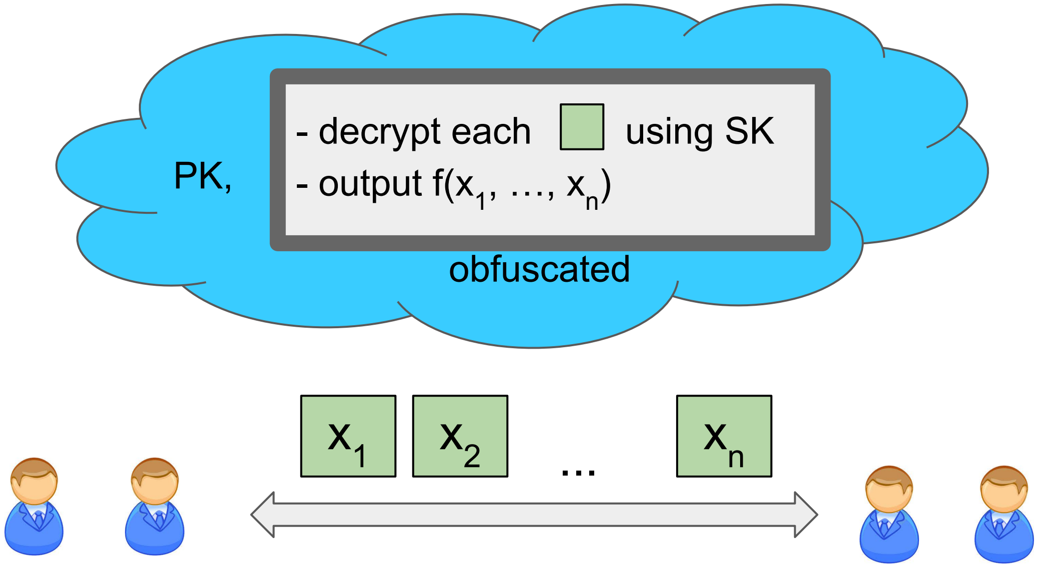
First attempt



First attempt

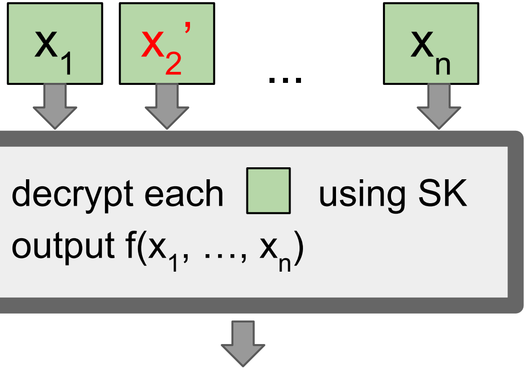
$\boxed{x_i} = \mathrm{Enc}_{PK}(x_i)$



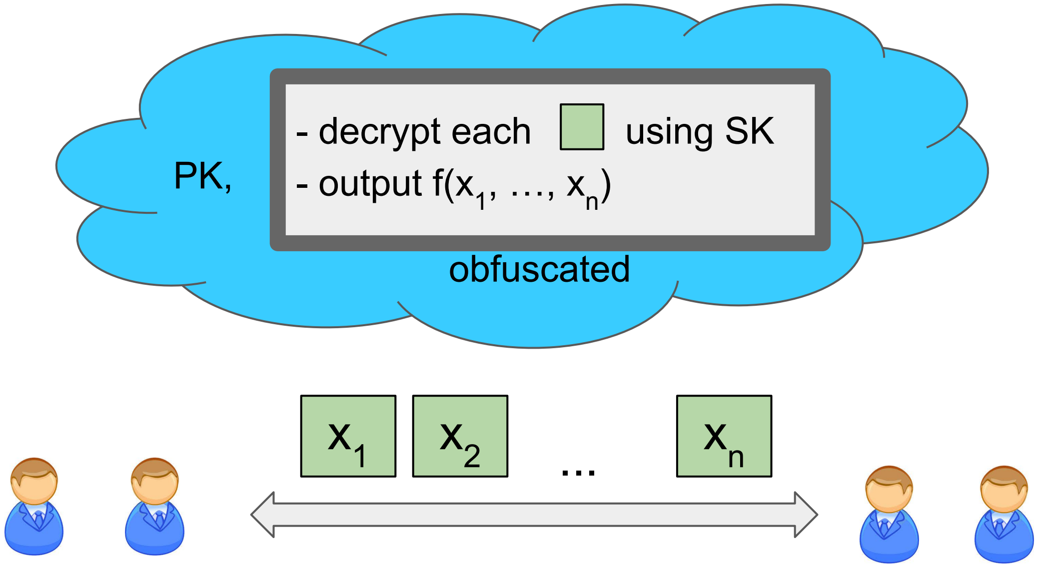


First attempt

$\boxed{x_i} = \mathrm{Enc}_{PK}(x_i)$



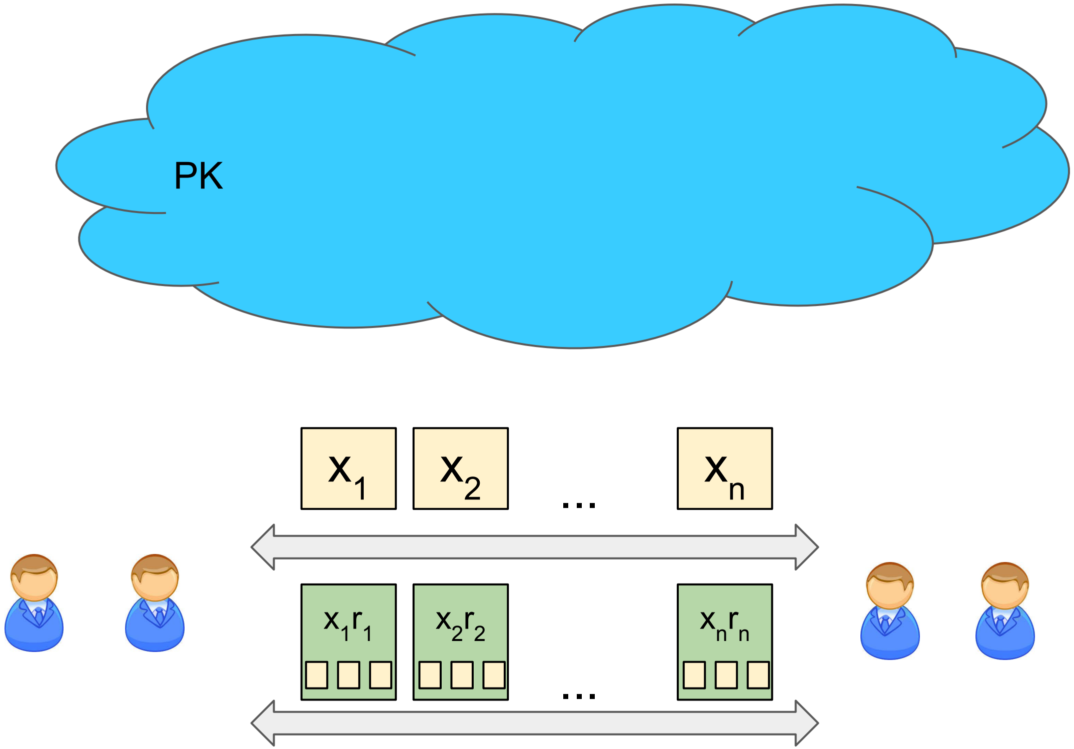
$y' = f(x_1, x_2'\ldots, x_n)$



Our protocol

$\boxed{x_i}$ = Commit(x_i; r_i)

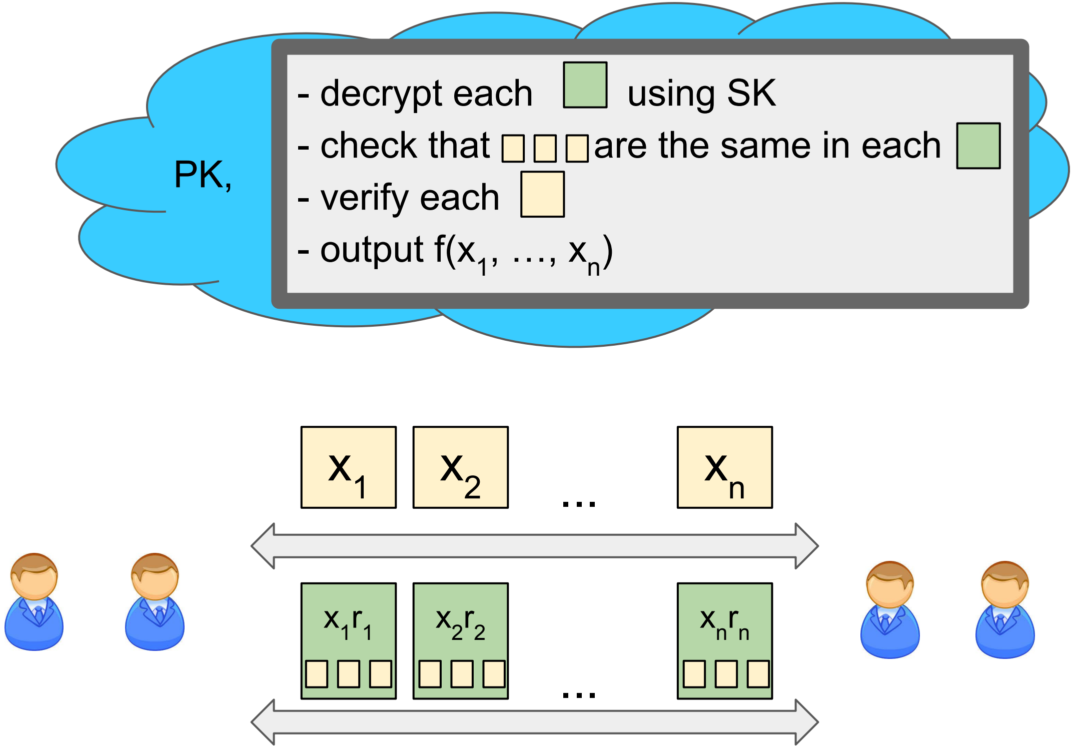
$\boxed{x_i r_i}$ = $\mathrm{Enc}_{PK}(x_i || r_i || \boxed{}\ \boxed{} \ldots \boxed{}\ )$



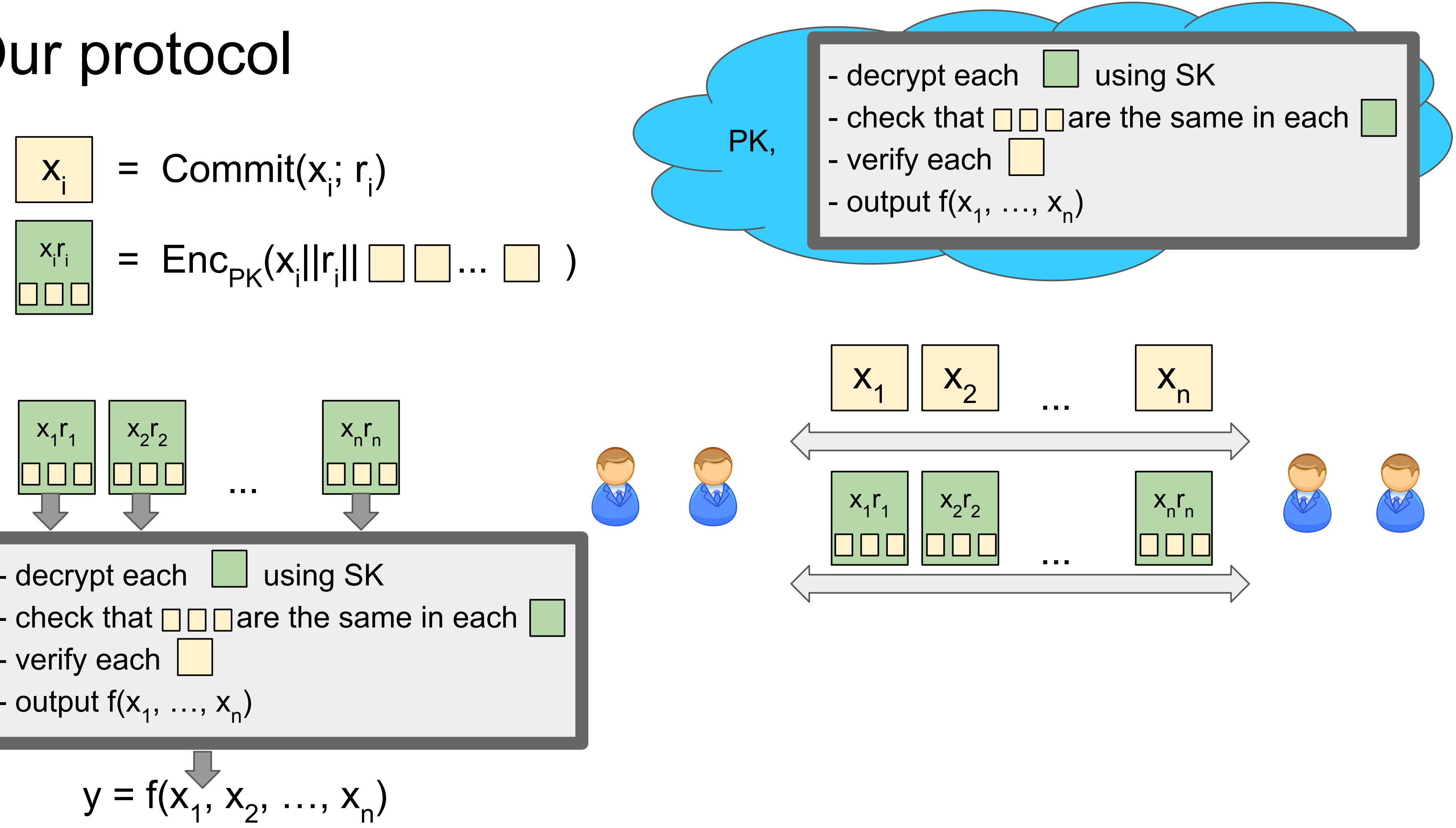
Our protocol

$\boxed{x_i} = \mathrm{Commit}(x_i; r_i)$

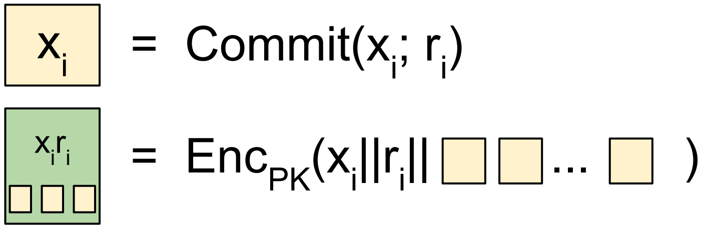
$\boxed{x_i r_i \; \square\square\square} = \mathrm{Enc}_{PK}(x_i || r_i || \square\, \square \ldots \square\ )$

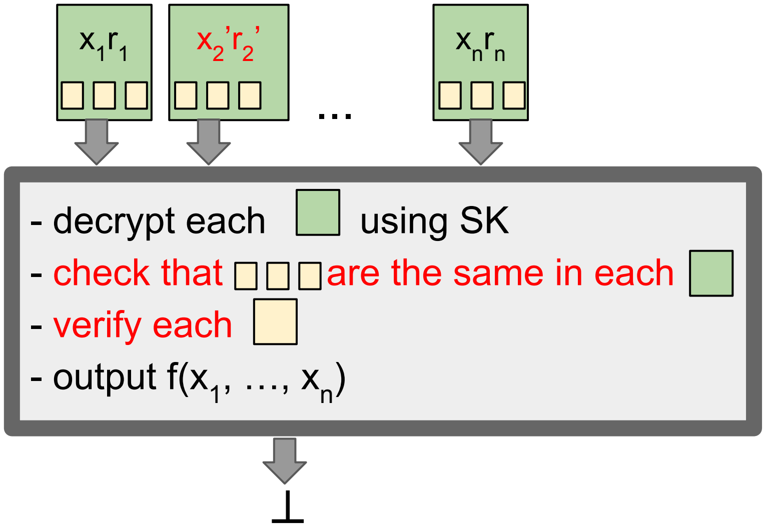


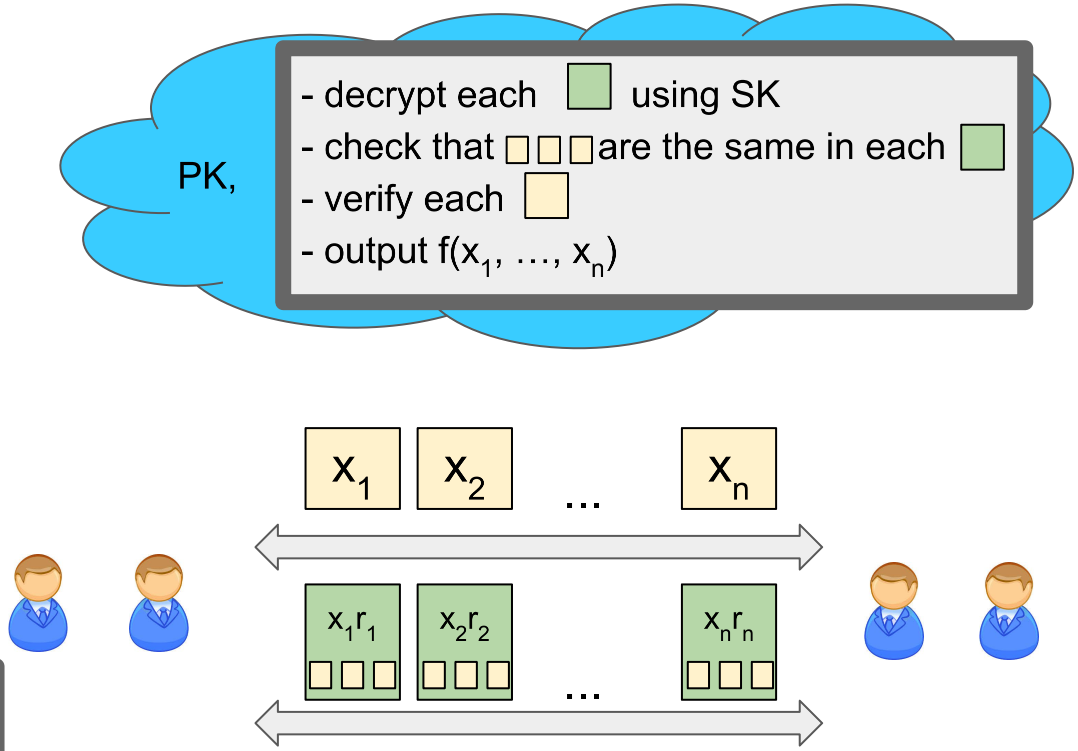
Our protocol



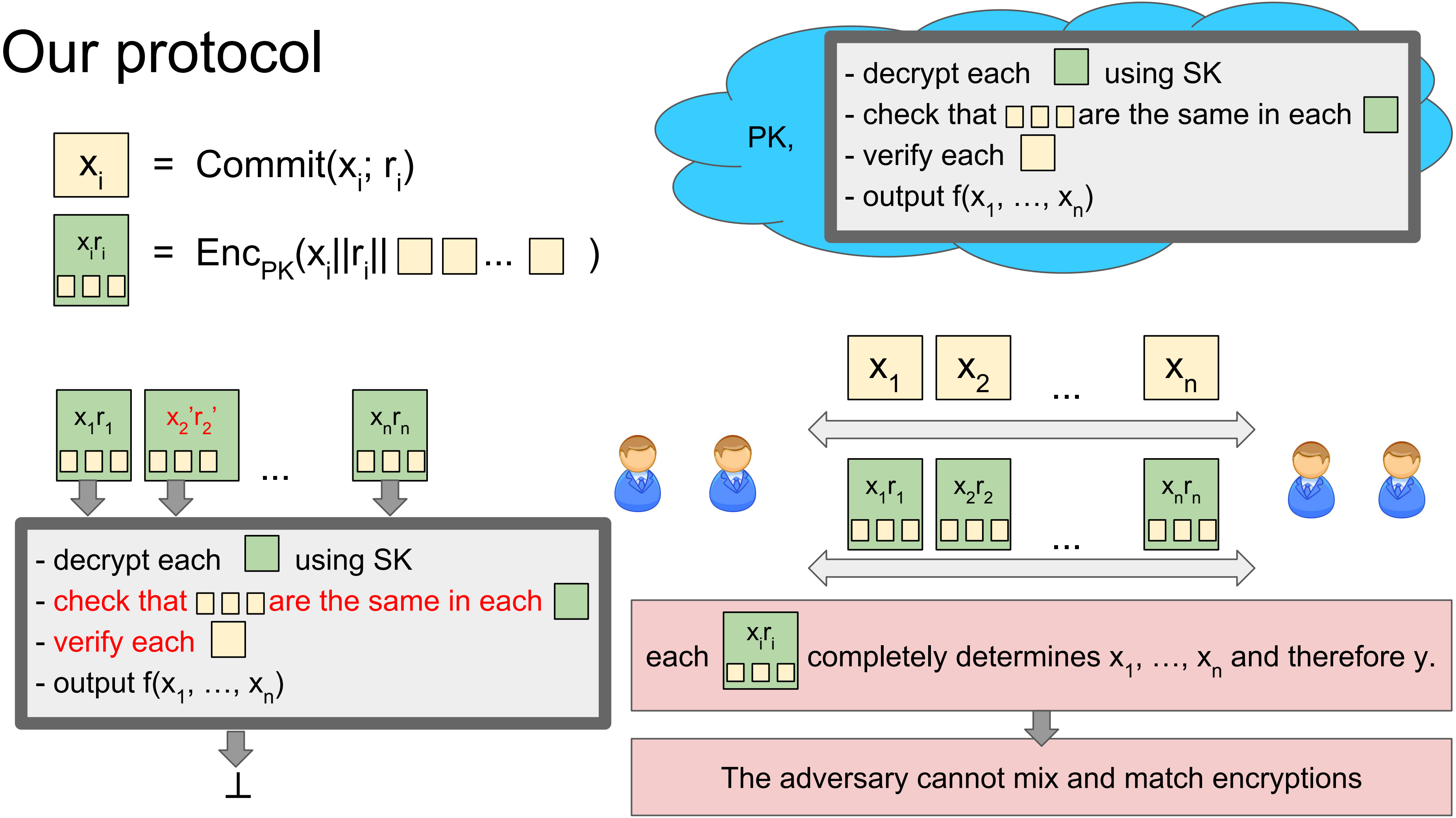
Our protocol



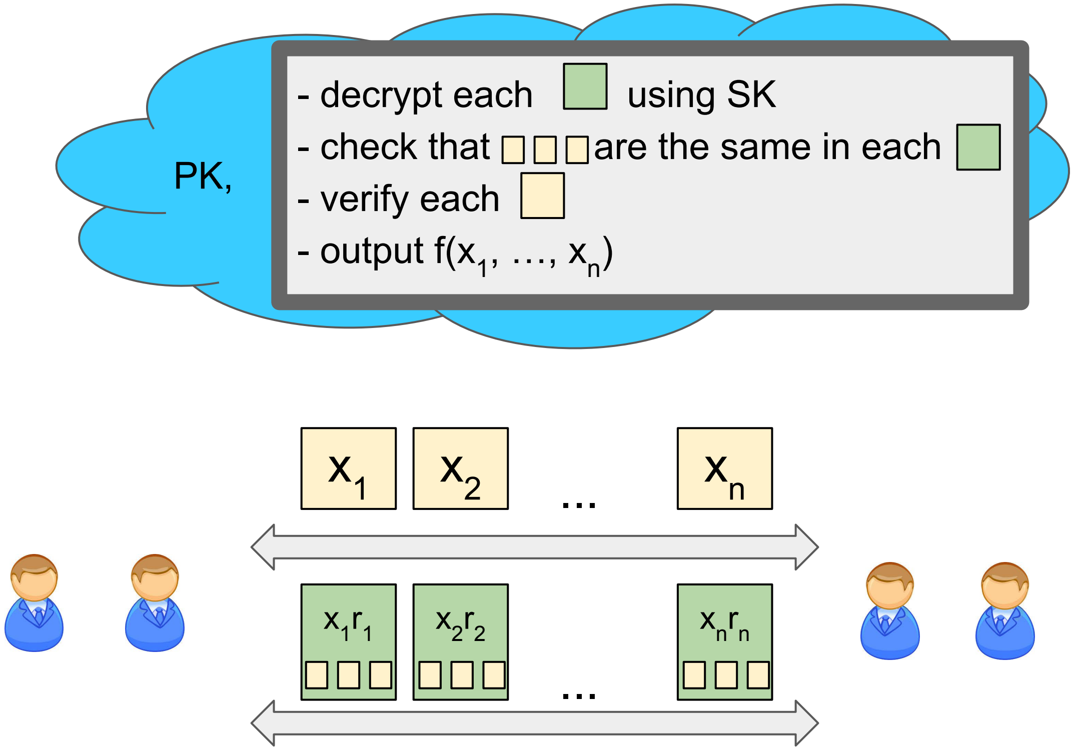




Our protocol



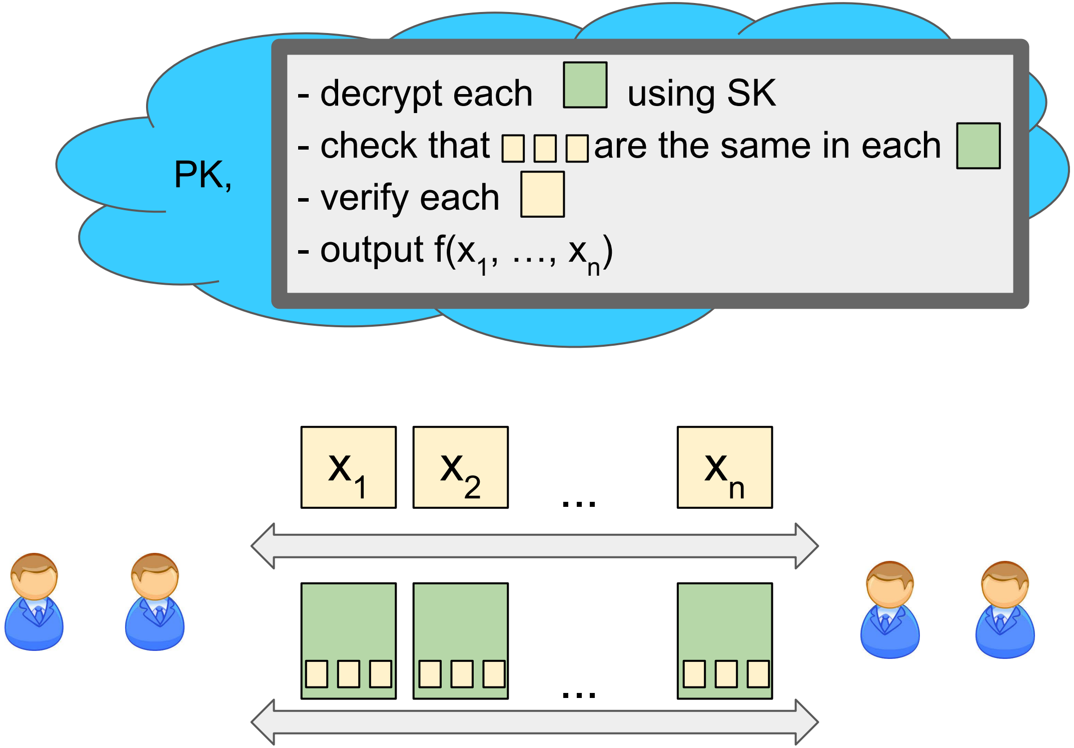
Wanted: Encryption



Problem:
cannot use security of encryption
since SK is in the program

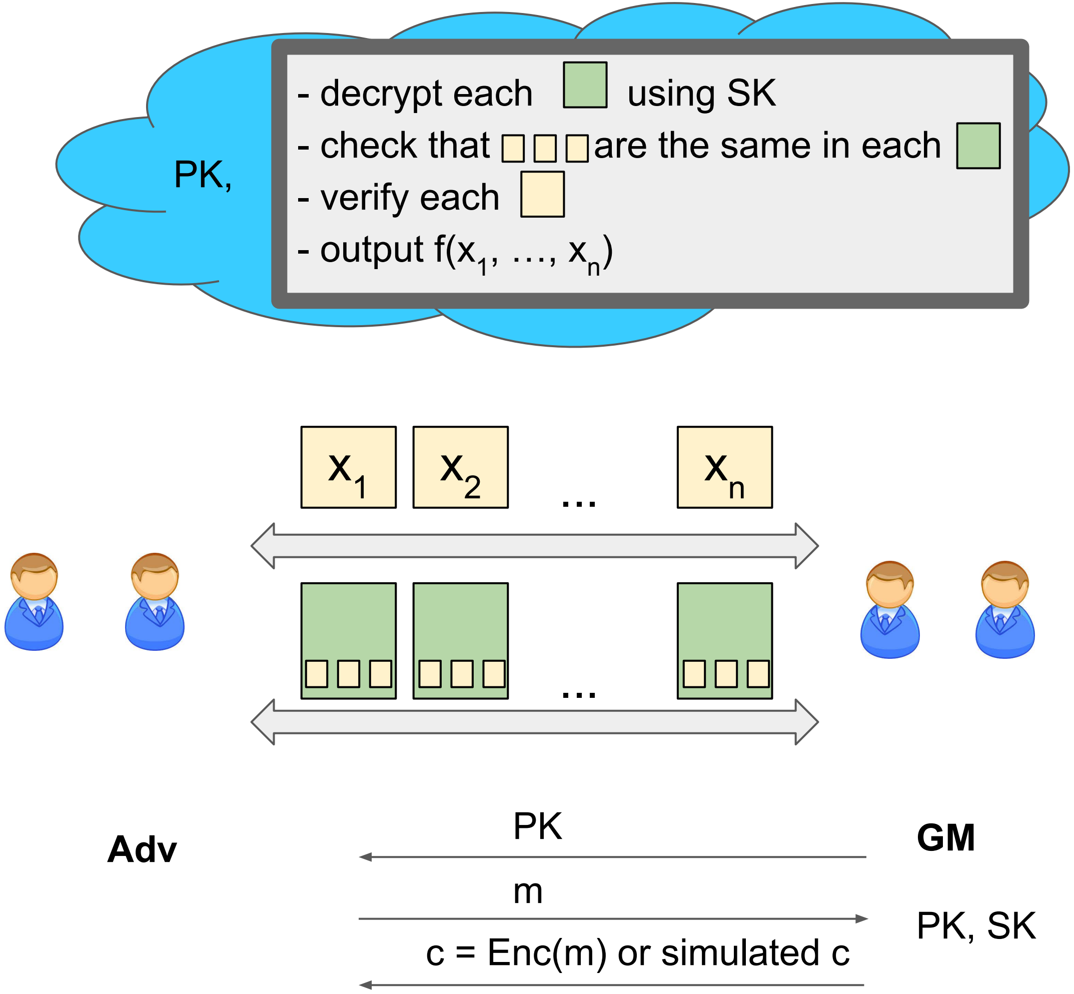
Wanted: Encryption

Problem:
cannot use security of encryption
since SK is in the program



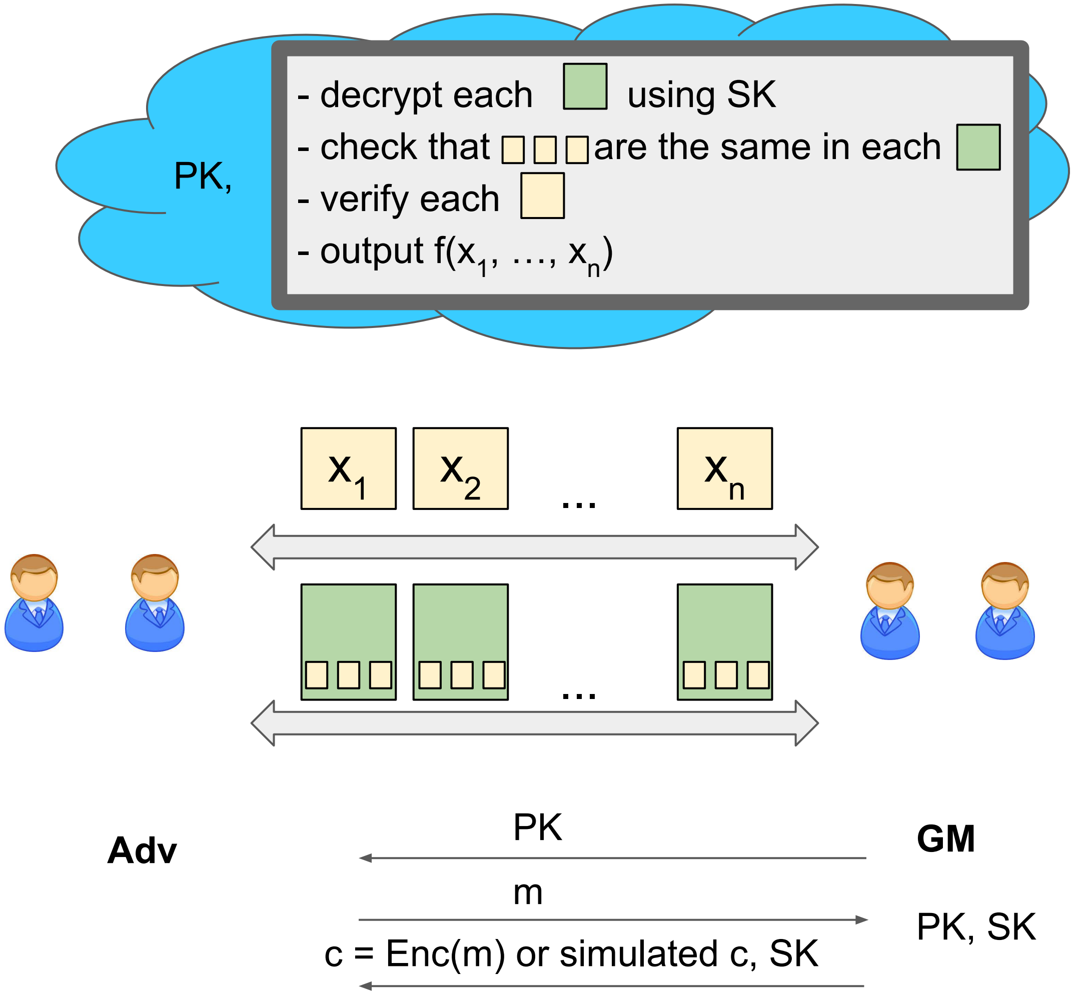
Wanted: Encryption

Problem:
cannot use security of encryption
since SK is in the program



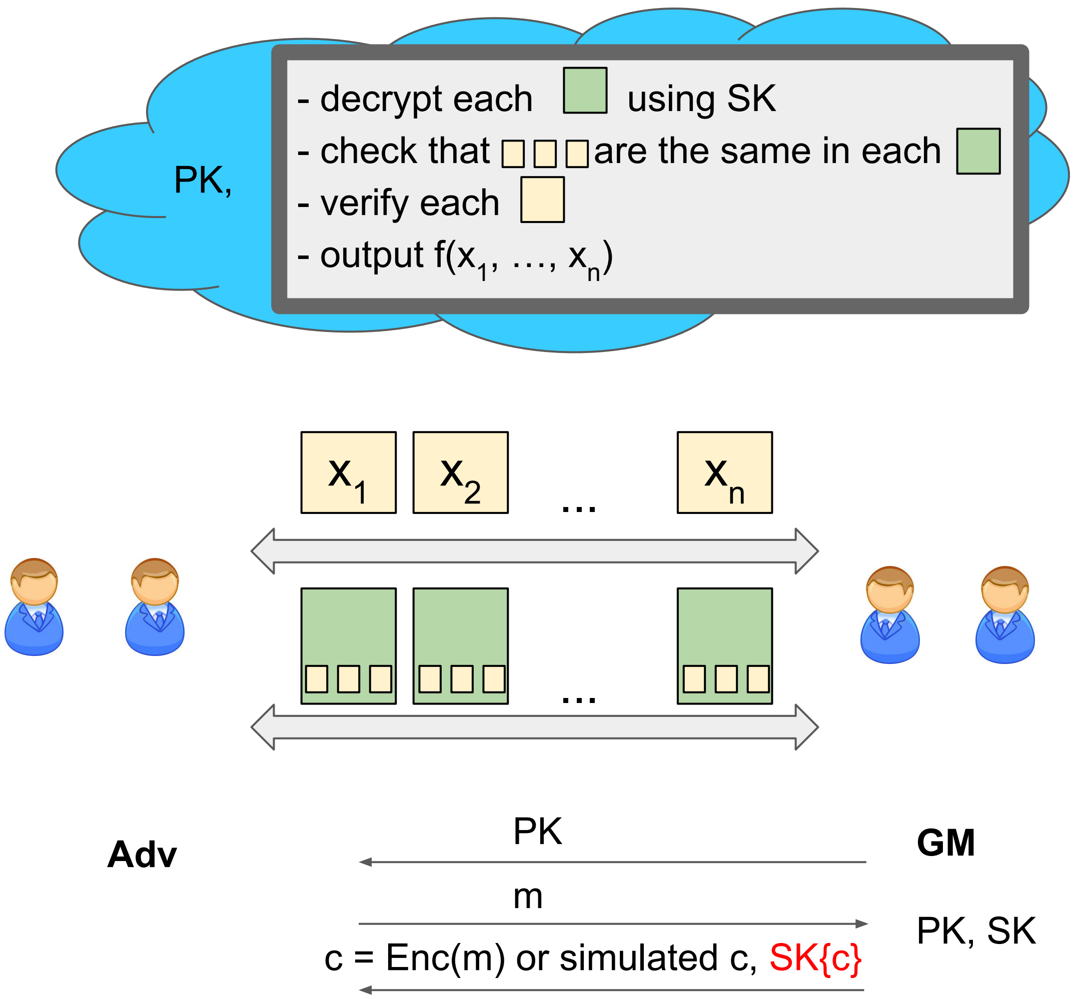
Wanted: Encryption

Problem:
cannot use security of encryption
since SK is in the program



Wanted: Encryption

Problem:
cannot use security of encryption
since SK is in the program

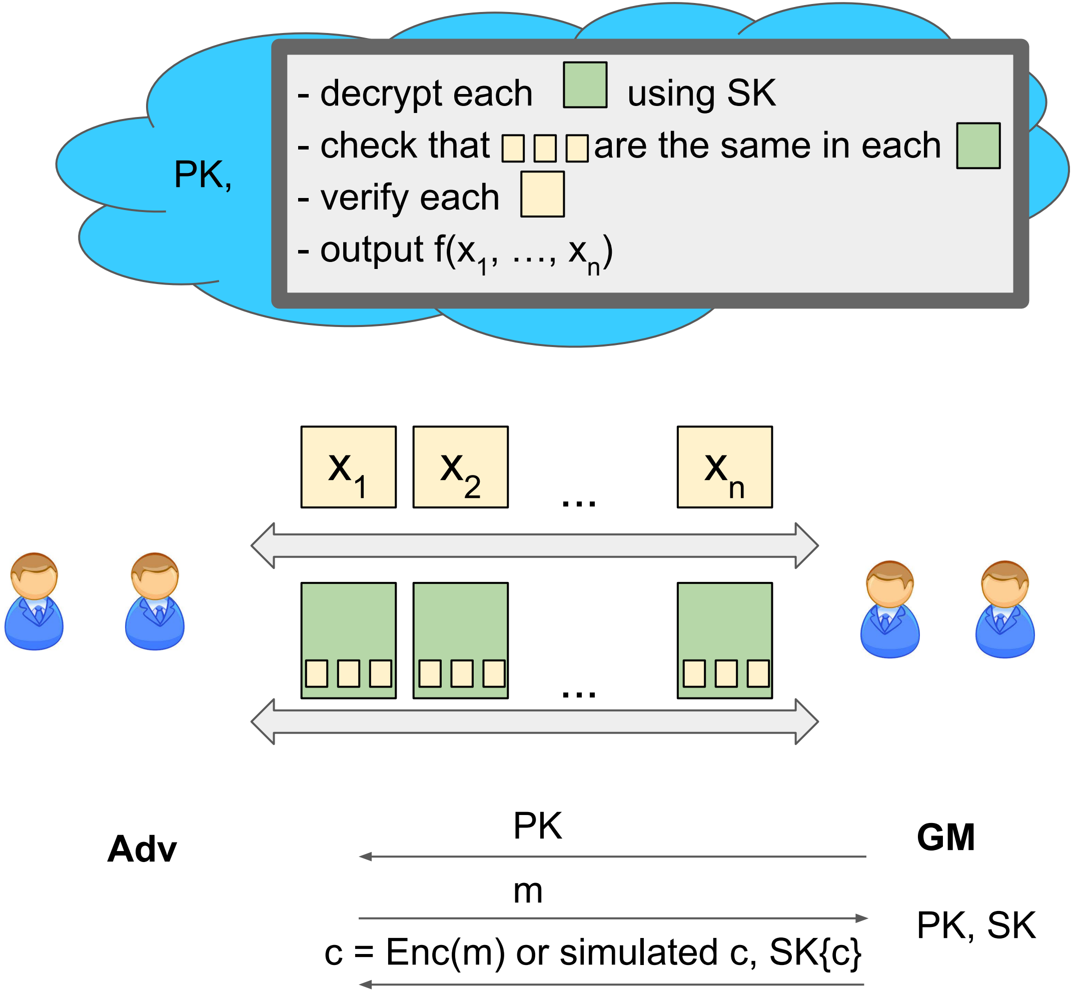


PRE

Problem:
cannot use security of encryption
since SK is in the program

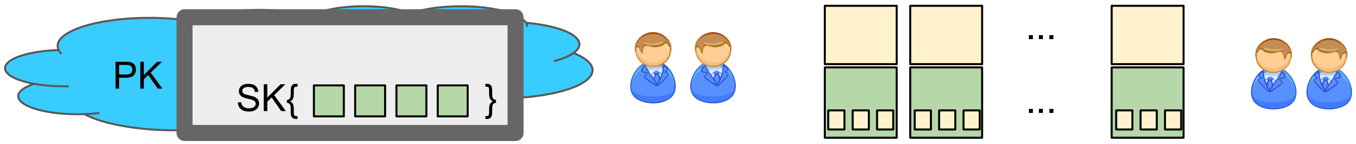
Solution:
Puncturable randomized encryption (PRE)
(from iO and OWFs)

Property:
simulation-secure
even when almost all SK is known



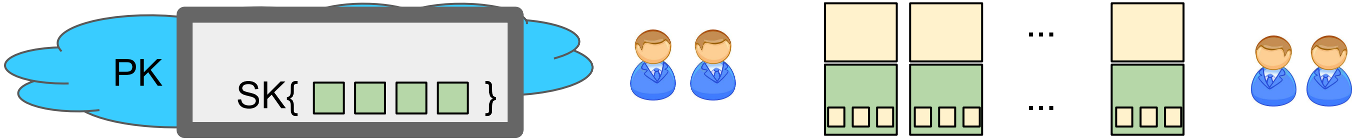
Achieving globality and full adaptive security

Simulation: not global



Achieving globality and full adaptive security

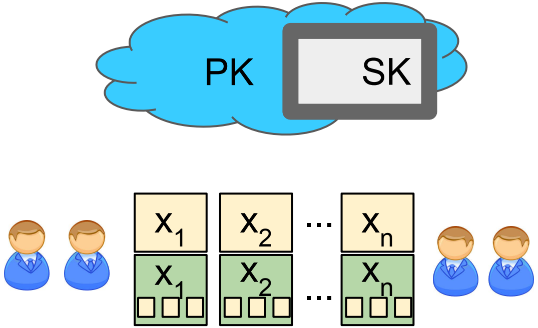
Simulation: not global



Solution: Modify the protocol to sample PK, 

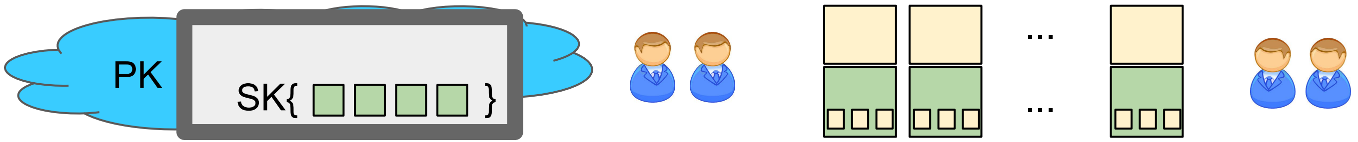
during the execution.

KSW'14
CPR'16

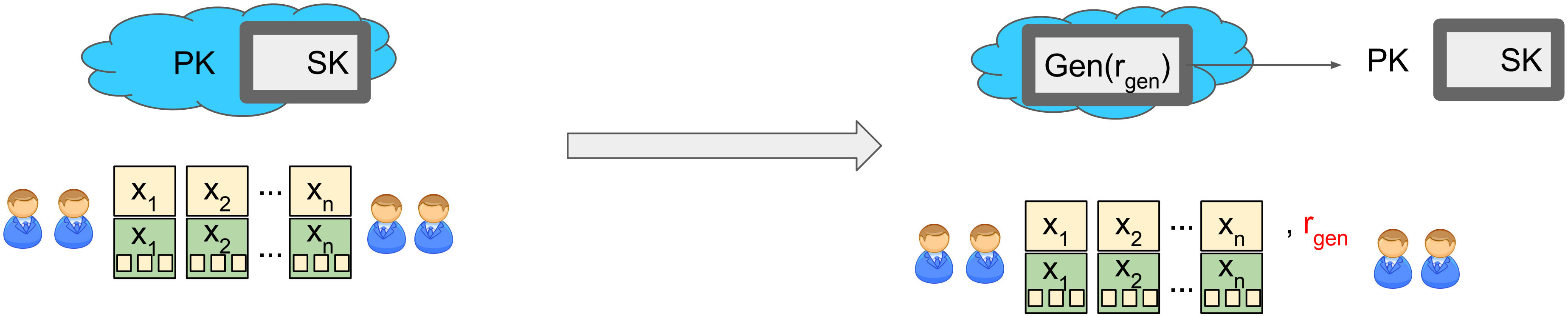


Achieving globality and full adaptive security

Simulation: not global



Solution: Modify the protocol to sample PK, SK during the execution.



How to make the protocol RAM-efficient

Ishai-Kushilevitz paradigm:
use MPC to evaluate garbling:
$F(x_1, \ldots, x_n; r)$ = garbled f, garbled $x_1, \ldots, x_n$.

How to make the protocol RAM-efficient

Ishai-Kushilevitz paradigm:
use MPC to evaluate garbling:
$F(x_1, \ldots, x_n; r) =$ garbled f, garbled $x_1, \ldots, x_n$.

Any MPC protocol + RAM-efficient garbling (e.g. CH'16) = RAM-efficient protocol

How to make the protocol RAM-efficient

Ishai-Kushilevitz paradigm:
use MPC to evaluate garbling:
$F(x_1, \ldots, x_n; r)$ = garbled f, garbled $x_1, \ldots, x_n$.

Any MPC protocol + RAM-efficient garbling (e.g. CH'16) = RAM-efficient protocol

Only works for n-1 corruptions!

How to make the protocol RAM-efficient

Ishai-Kushilevitz paradigm:
use MPC to evaluate garbling:
$F(x_1, \ldots, x_n; r)$ = garbled f, garbled $x_1, \ldots, x_n$.

Any MPC protocol + RAM-efficient garbling (e.g. CH'16) = RAM-efficient protocol

Only works for n-1 corruptions!
For full adaptive security:

randomness-hiding MPC protocol + RAM-efficient garbling (e.g. CH'16) = RAM-efficient protocol

Our results :

Semi-honest case

2 round fully adaptive MPC with nice properties (randomness-hiding, RAM-efficient, global CRS...)

malicious case

NIZK with RAM efficiency

plug into GP’15

GP’15 2 round fully adaptive MPC becomes **RAM-efficient**

Part II: Making GP’15 RAM-efficient

Part II: Making GP’15 RAM-efficient

Any
randomness-hiding
MPC protocol

+

RAM-efficient
garbling
(e.g. CH’16)

=

RAM-efficient
protocol

GP’15 doesn’t hide randomness

Malicious case: achieving RAM-efficiency

Theorem (Garg-Polychroniadou'15):
subexponential IO+TDPs → malicious MPC (2 round, fully adaptive)

Malicious case: achieving RAM-efficiency

Theorem (Garg-Polychroniadou'15):
subexponential IO for RAM +TDPs → malicious MPC for RAM? (2 round, fully adaptive)

Malicious case: achieving RAM-efficiency

didn't have before...

Theorem (Garg-Polychroniadou'15):
subexponential IO for RAM +TDPs + statistically-sound NIZK for RAM
→ malicious MPC for RAM (2 round, fully adaptive)

Google

zero knowledge ram

Press Enter to search.

Google

zero knowledge ram

Press Enter to search.

RAM-efficient NIZK

```
f(x):
For i = 1… 100000000 do {
}
```

RAM-efficient NIZK

```
f(x):
For i = 1… 100000000 do {
}
```

- $|\text{proof}| \sim |f|_{RAM}$

Prior work on RAM-efficient NIZK

```
f(x):
For i = 1… 100000000 do {
}
```

- |proof| ~ $|f|_{RAM}$ - **done**

[Gen09, Gro11]:
- |proof| ~|w|

Prior work on RAM-efficient NIZK

```
f(x):
For i = 1… 100000000 do {
}
```

- $|proof| \sim |f|_{RAM}$ - **done**

[Gen09, Gro11]:
- |proof| ~|w|
- Verify ~ circuit compexity of f

Obfuscated program in GP'15:

```
Verify proof for "f(x_1..x_n)=y,..."
...
...
```

Prior work on RAM-efficient NIZK

```
f(x):
For i = 1… 100000000 do {
}
```

- $|proof| \sim |f|_{RAM}$ - **done**
- Verification complexity ~ RAM complexity of f - **?**

[Gen09, Gro11]:
- $|proof| \sim |w|$
- Verify ~ circuit compexity of f

Obfuscated program in GP’15:

```
Verify proof for “f(x_1..x_n)=y,...”
...
...
```

Malicious case

Theorem (Garg-Polychroniadou'15):
subexponential IO for RAM + TDPs+ statistically-sound NIZK for RAM
→ malicious MPC for RAM (2 round, fully adaptive)

Theorem (Our work):
Garbled RAM + NIZK proofs for circuits → statistically-sound NIZK for RAM.

Malicious case

Theorem (Garg-Polychroniadou'15):
subexponential IO for RAM + TDPs+ statistically-sound NIZK for RAM
→ malicious MPC for RAM (2 round, fully adaptive)

Theorem (Our work):
Garbled RAM + NIZK proofs for circuits → statistically-sound NIZK for RAM.

Corollary:
Subexp. iO+TDPs → malicious MPC for RAM (2 round, fully adaptive)

NIZK + Garbled RAM → NIZK for RAM

Attempt 1

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

Verifier

x ∈ L

NIZK + Garbled RAM → NIZK for RAM

Attempt 1

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

Verifier

x ∈ L

R → R(*,*)

x, w → x, w

garbled RAM:
- allows to compute R(x; w)
- hides R, x, w
- RAM-efficient

NIZK + Garbled RAM → NIZK for RAM

Attempt 1

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

Proof π = R(*,*) x, w →

Verifier

x ∈ L

Accept if Eval(R(*,*) x, w) = 1

R → R(*,*)

x, w → x, w

garbled RAM:

- allows to compute R(x; w)
- hides R, x, w
- RAM-efficient

NIZK + Garbled RAM → NIZK for RAM

Attempt 1

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

Proof π = R(*,*) x, w

Verifier

x ∈ L

Accept if Eval(R(*,*) x, w) = 1

R → R(*,*)
x, w → x, w

garbled RAM:
- allows to compute R(x; w)
- hides R, x, w
- RAM-efficient

- Verifier doesn't learn anything about w

NIZK + Garbled RAM → NIZK for RAM

Attempt 1

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

Proof $\boldsymbol{\pi}$ = R(*,*) x, w

Verifier

x ∈ L

R → R(*,*)

x, w → x, w

garbled RAM:
- allows to compute R(x; w)
- hides R, x, w
- RAM-efficient

Accept if Eval(R(*,*) x, w) = 1

- Verifier doesn't learn anything about w
- Malicious prover can garble R ≡ 1

NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that R(x; w) = 1

Prover

$x \in L$
w

R(*,*)   x, w

NIZK proof: “garbling done correctly, for correct R and x”

Verifier

$x \in L$

R → R(*,*)

x, w → x, w

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

R(*,*) x, w

NIZK proof: "garbling done correctly, for correct R and x"

short (~$|R|_{RAM}$)

Verifier

x ∈ L

R → R(*,*)

x, w → x, w

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

R(*,*)   x, w

NIZK proof: “garbling done correctly, for correct R and x”

Verifier

x ∈ L

R → R(*,*)
x, w → x, w

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

- Verifier doesn’t learn anything about w
-

NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that $\exists$ w such that R(x; w) = 1

Prover

x ∈ L
w

R(*,*)   x, w

NIZK proof: "garbling done correctly, for correct R and x"

Verifier

x ∈ L

R → R(*,*)

x, w → x, w

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

- Verifier doesn't learn anything about w
- garbling: for **most** random coins of garbling, correctness holds.

NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

R(*,*) | x, w

NIZK proof: “garbling done correctly, for correct R and x”

Verifier

x ∈ L

R → R(*,*)
x, w → x, w

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

- Verifier doesn’t learn anything about w
- garbling: for **most** random coins of garbling, correctness holds.
- what if for some r garbling always garbles R ≡ 1

NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

R(*,*) | x, w

NIZK proof: “garbling done correctly, for correct R and x”

Verifier

x ∈ L

R → R(*,*)
x, w → x, w

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

- Verifier doesn’t learn anything about w
- No coin tossing - need **perfectly correct** garbled RAM
- Currently do not have garbled RAM with perfect correctness

NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that R(x; w) = 1

Prover

$x \in L$
w

R(*,*) | x, w

NIZK proof: "garbling done correctly, for correct R and x"

Verifier

$x \in L$

R → R(*,*)
x, w → x, w

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

- Verifier doesn't learn anything about w
- CH15 garbled RAM satisfies **perfect correctness with abort** - enough

NIZK + Garbled RAM → NIZK for RAM

Attempt 2

Convince that ∃ w such that R(x; w) = 1

Prover

x ∈ L
w

R(*,*)   x, w

NIZK proof: “garbling done correctly, for correct R and x”

→

Verifier

x ∈ L

R → R(*,*)

x, w → x, w

Accept if Eval(R(*,*) x, w) = 1

and if NIZK verifies.

- Verifier doesn’t learn anything about w
- CH15 garbled RAM satisfies **perfect correctness with abort** - enough
- evaluator either gets **correct** output, or **rejects**

Summary: two round adaptively secure protocols

Semi-honest case:

- global CRS
- supports RAM
- randomness-hiding (e.g. N = pq)

Malicious case (GP15 + our RAM efficient NIZK):

- RAM-efficient

Questions?