Efficient Leakage Resilient Circuit Compilers

Marcin Andrychowicz, Warsaw University, Poland

Ivan Damgård, Aarhus University, Denmark

Stefan Dziembowski, Warsaw University, Poland

Sebastian Faust, EPFL, Switzerland

Antigoni Polychroniadou, Aarhus University, Denmark

Theory

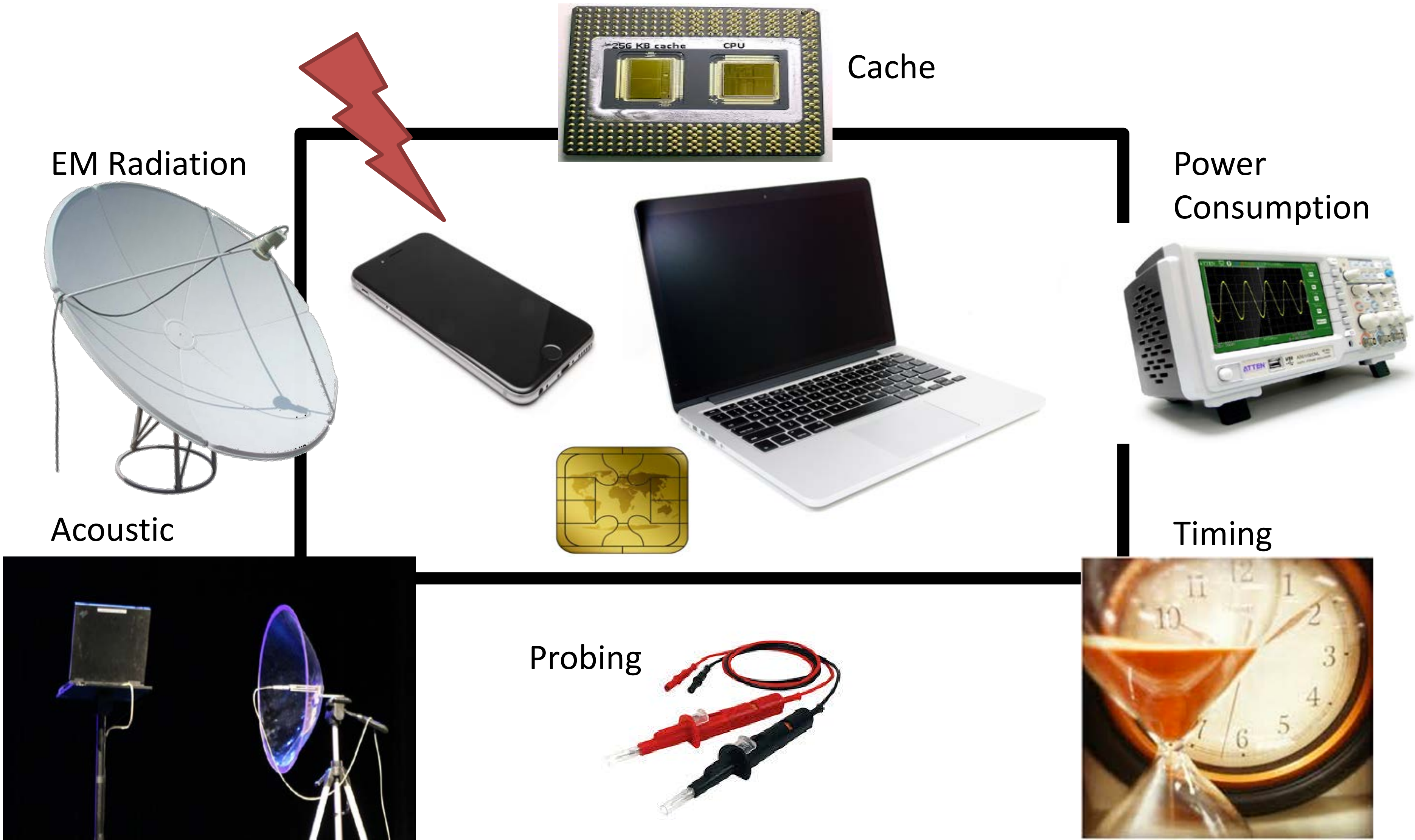
Cryptographic algorithms are often modeled as 'black boxes'

E.g. Internal computation is opaque to external adversaries.

Security is proven under various hardness assumptions.

Reality

Computation Internals **Leak**



Motivation

Many provably secure cryptosystems can be broken by side-channel attacks

Two Paradigms to Fight Leakage Attacks

1. Consider Leakage at design level

   Only security of specific schemes.

How to securely implement any scheme?

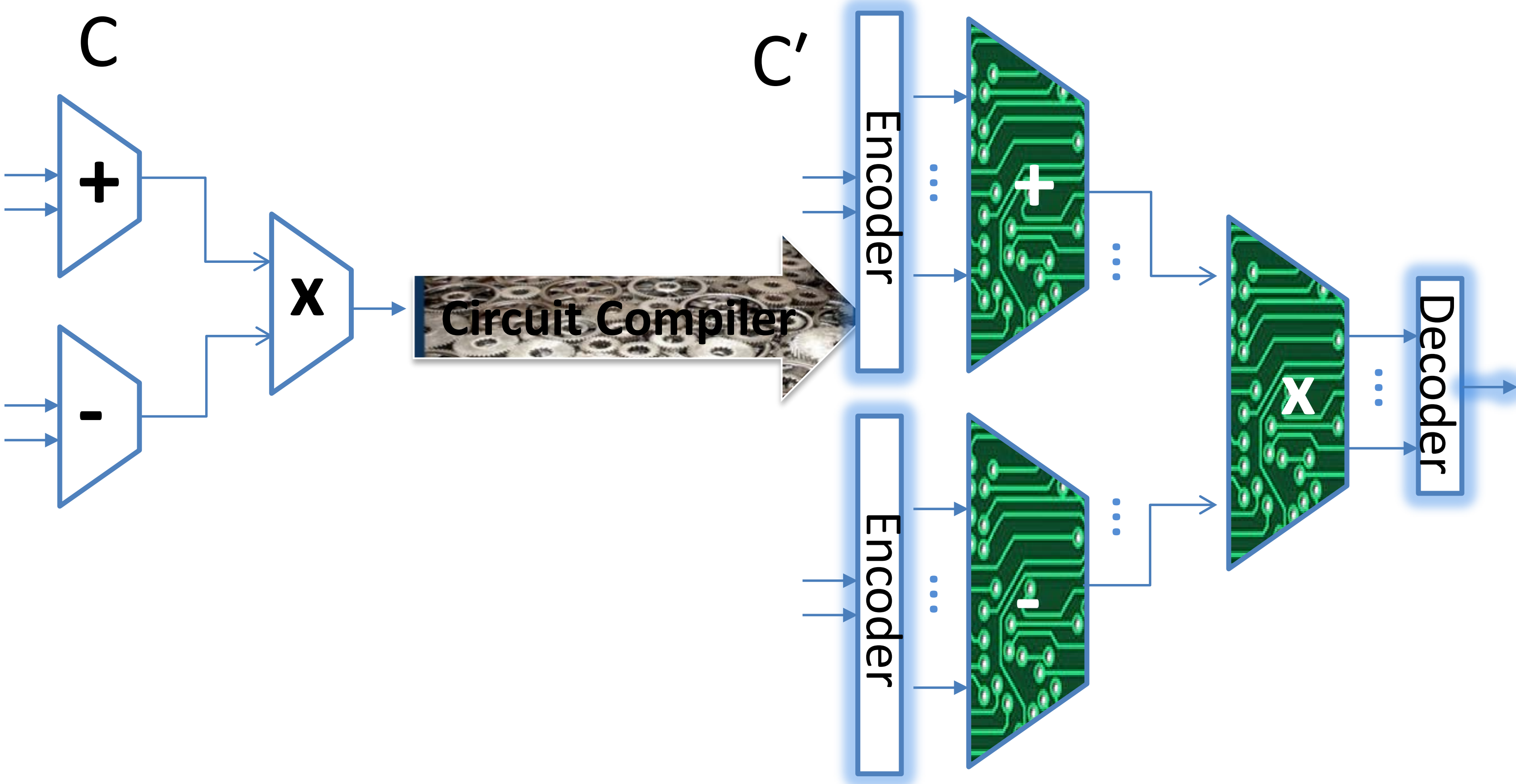
Wanted:

2. Leakage resilient Compiler

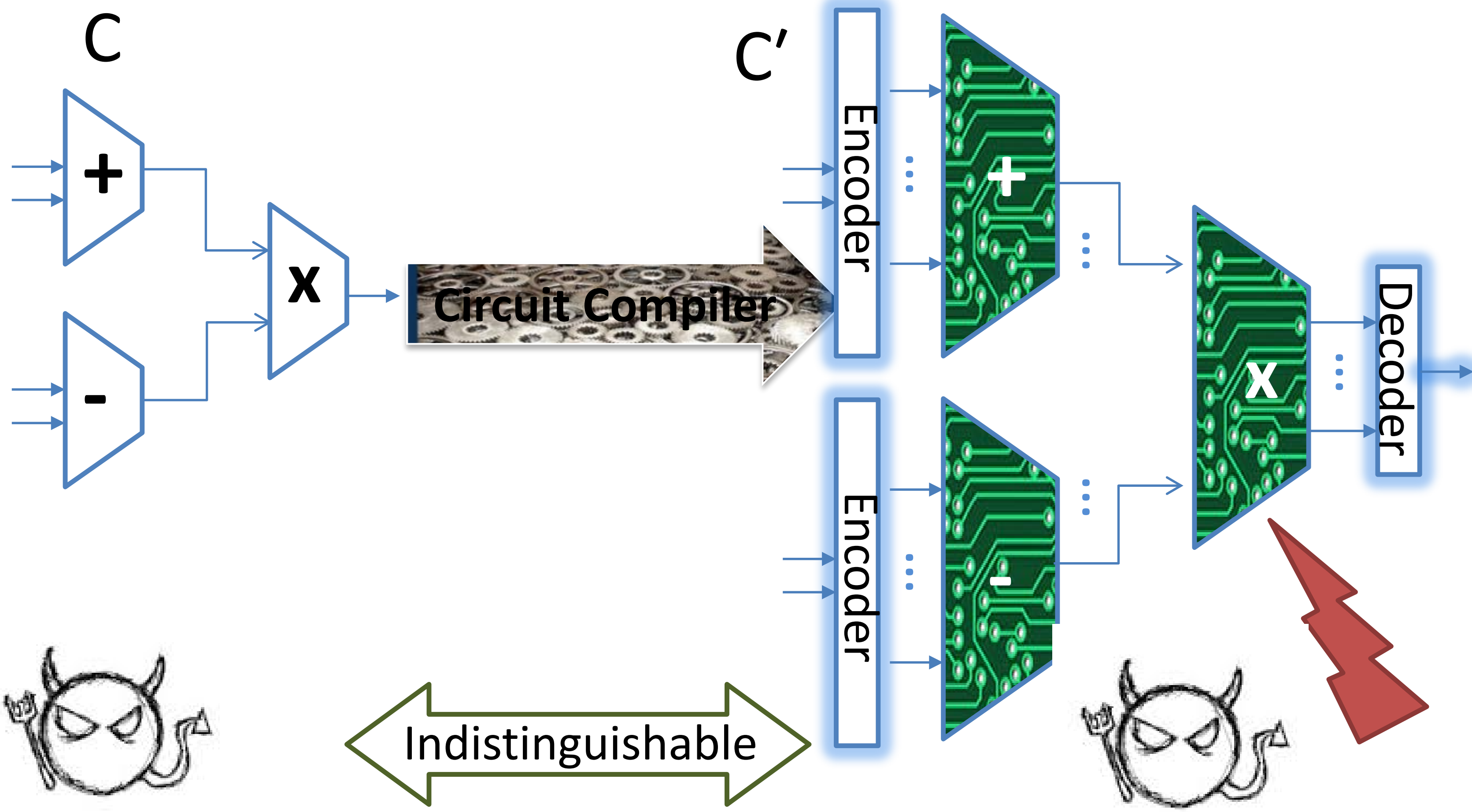
   Transform any circuit to a leakage resilient circuit secure in a strong black-box sense.

Def.: Circuit Compilers

Def.: Circuit Compilers

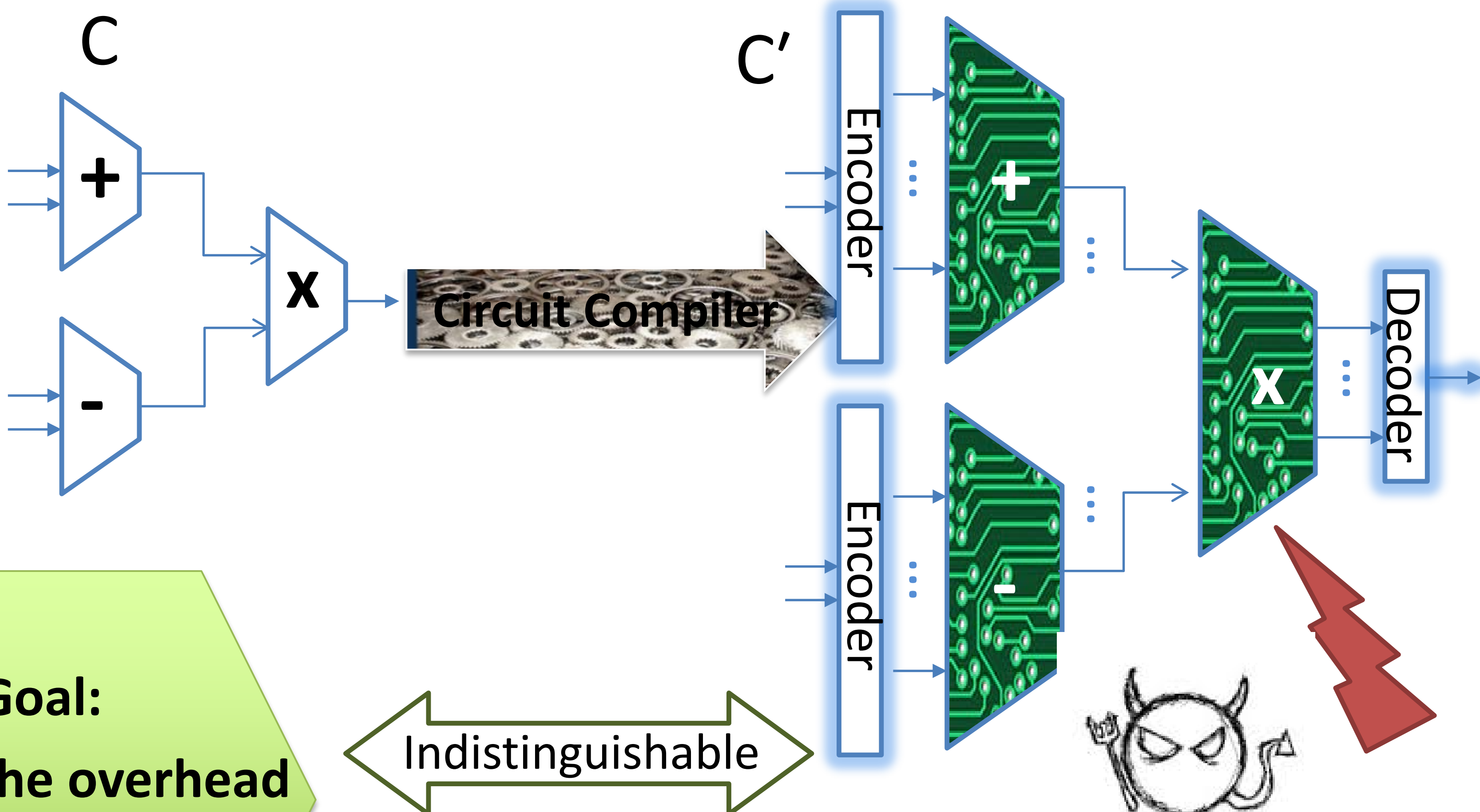


Def.: Circuit Compilers



Even given leakage, execution “looks like” black-box access to C(.)

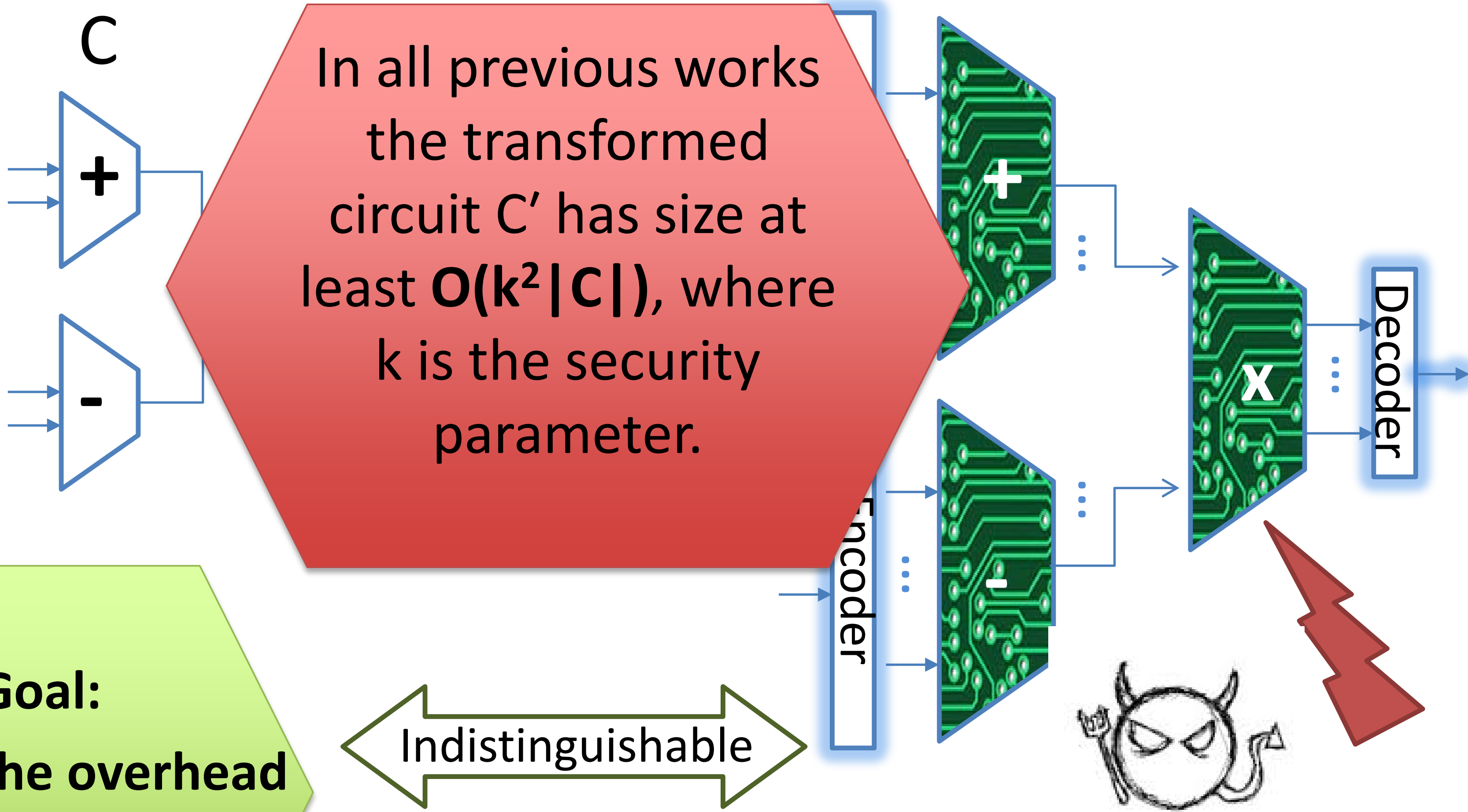
Def.: Circuit Compilers



Goal:
Reduce the overhead induced by the compiler

Even given leakage, execution "looks like" black-box access to C(.)

Def.: Circuit Compilers



In all previous works the transformed circuit C′ has size at least **$O(k^2|C|)$**, where k is the security parameter.

Goal:
Reduce the overhead induced by the compiler

Indistinguishable

Even given leakage, execution “looks like” black-box access to C(.)

Our Goal

Build Efficient Leakage Resilient Compilers

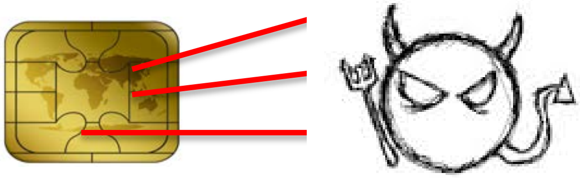
Is it possible to construct leakage resilient compilers with at most linear overhead?

- All previous works introduce at least quadratic overhead.

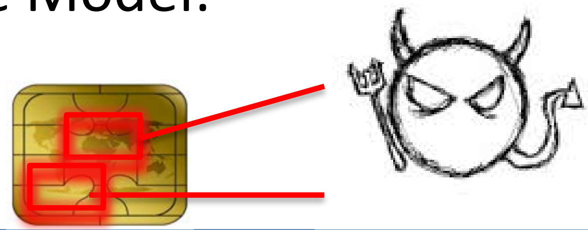
Prior Work on General Compilers

Three Leakage Models:

‘Local’ Bounded Wire-Probing: [ISW03,…]



‘Local’ Only Computation (OC) Leakage/ Split State Model: [MR04,…]



‘Global’ Computational Continuous Weak Leakage i.e. AC^0 leakage Functions [FRRTV10,…]

Our Results

Using Techniques from secure MPC

Efficient Compliers:

'Local' Wire-Probing: **$O(\text{polylog}(k) \cdot |C| \log |C|)$**
Previous Best Overhead: $O(k^2|C|)$ by [ISW03]

'Local' OC Leakage : **$O(k \log k \log\log k |C|)$**

Previous Best Overhead: $\Omega(k^4|C|)$ by [DF12] and $\Omega(k^3|C|)$ by [GR12]

This talk

'Global' Computational Continuous Weak Leakage: **$O(k \cdot |C| \log |C|)$**
Previous Best Overhead: $O(k^2|C|)$ by [FRRTV10] and $O(k^3|C|)$ by [R13]

Our Result on Global Computational Weak Leakage

- **Informal Theorem :** A compiler that makes **any circuit** resilient to **computationally weak leakages**. The compiler increases the circuit size by a factor of **O(k)**.

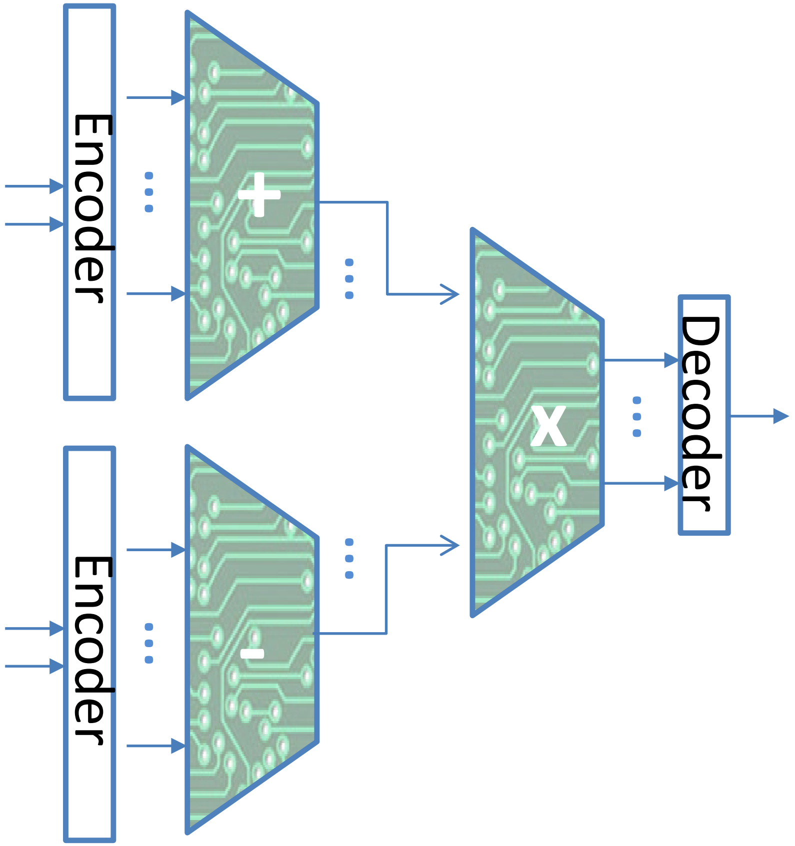
- **Global adaptive leakage**
- **Arbitrary total leakage**

However we must assume something [MR04]:

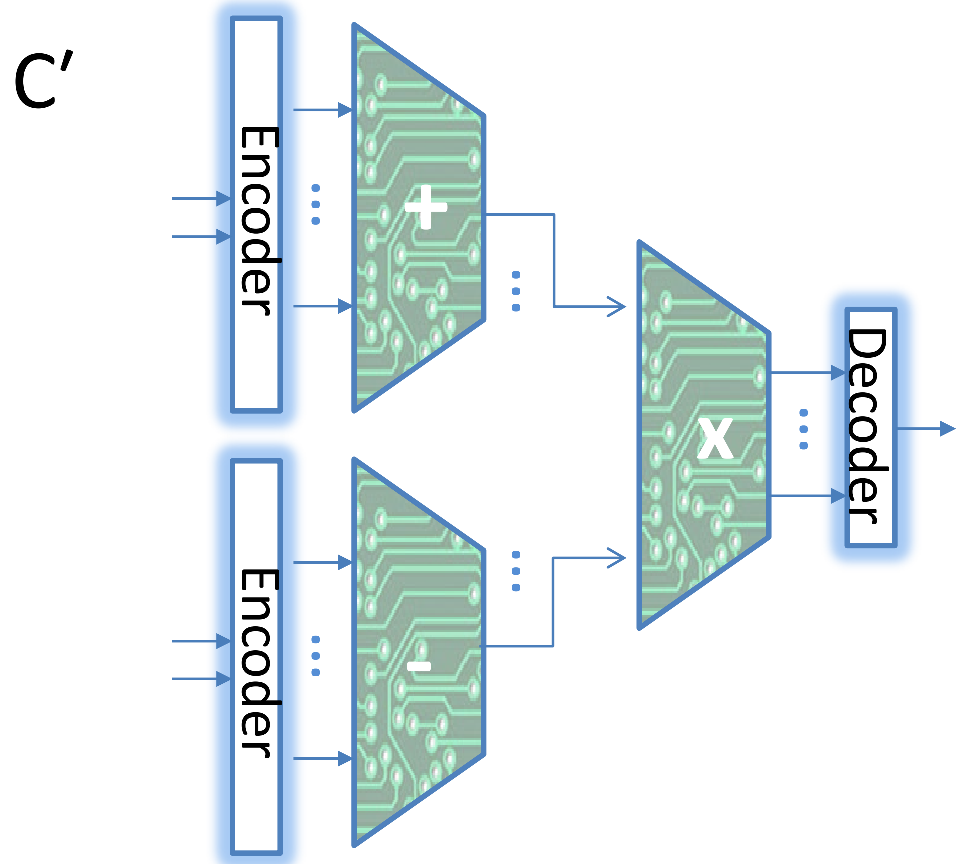
- **Leakage function is computationally weak.**
- **Simple opaque gates.**

The Compiler

C′



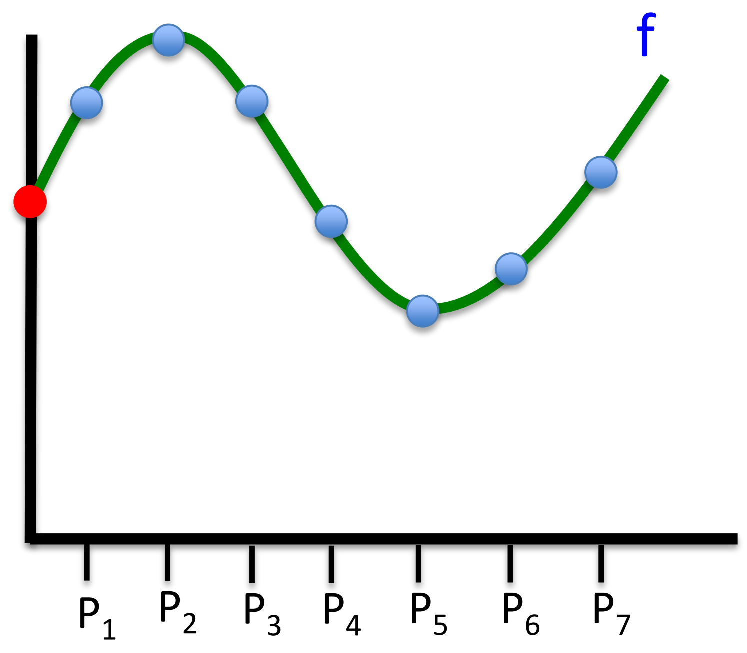
The Compiler:
From Wires to Wire Bundles



Packed Secret Sharing (PSS)

- PSS is a central tool in information theoretic secure MPC protocols.

Standard Secret Sharing :

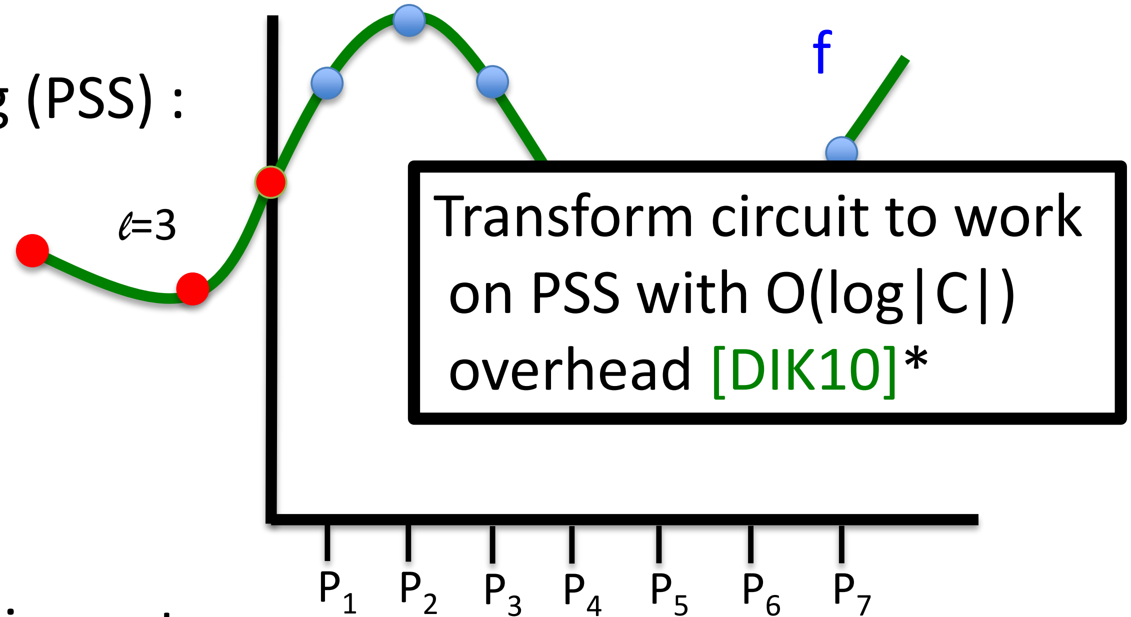


Degree of f denoted by d

Packed Secret Sharing (PSS)

- PSS is a central tool in information theoretic secure MPC protocols.

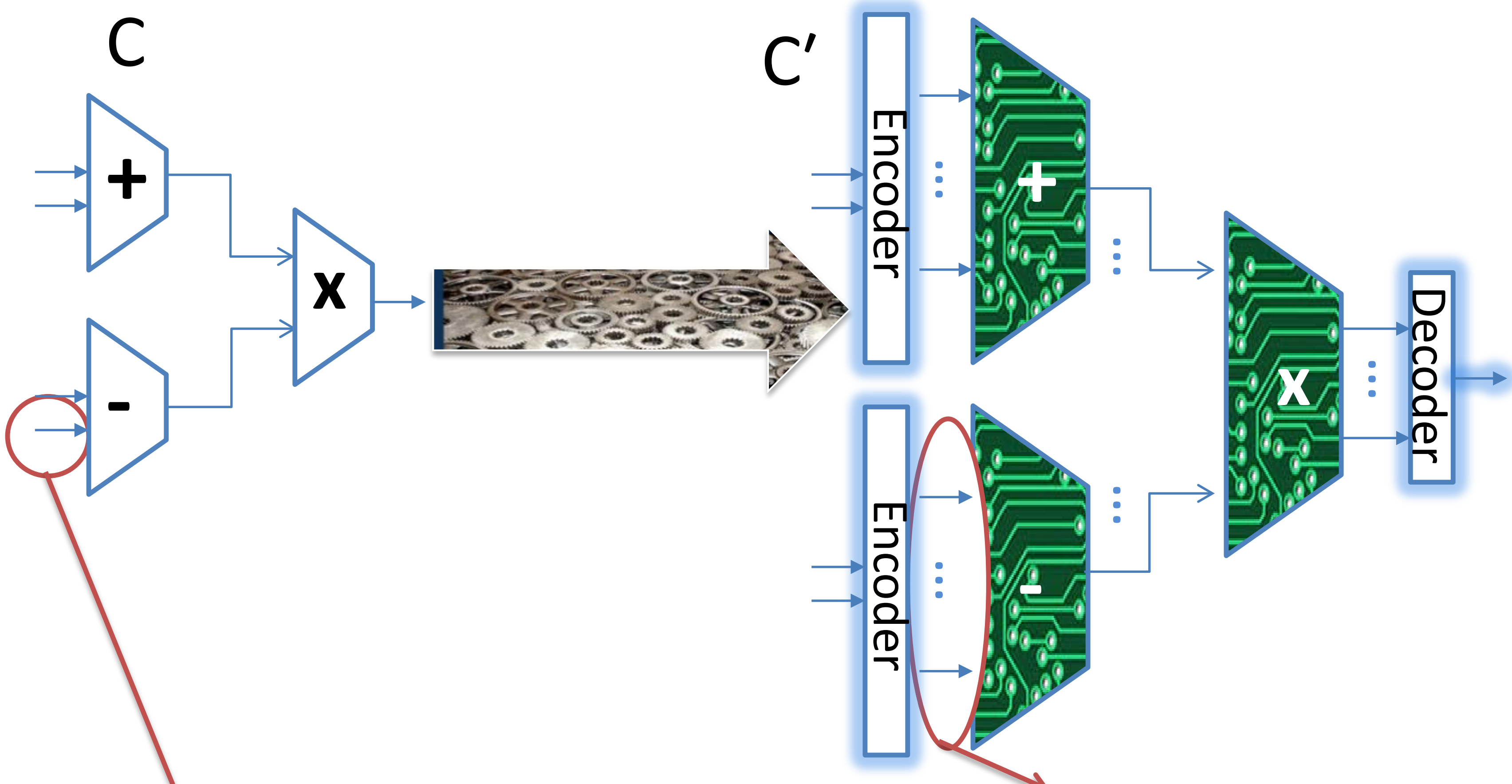
Packed Secret Sharing (PSS) :



Transform circuit to work on PSS with O(log|C|) overhead [DIK10]*

*Introduces Permutation gates.

- Every wire is encoded with PSS.
- Inputs are encoded; outputs are decoded.



Each wire w

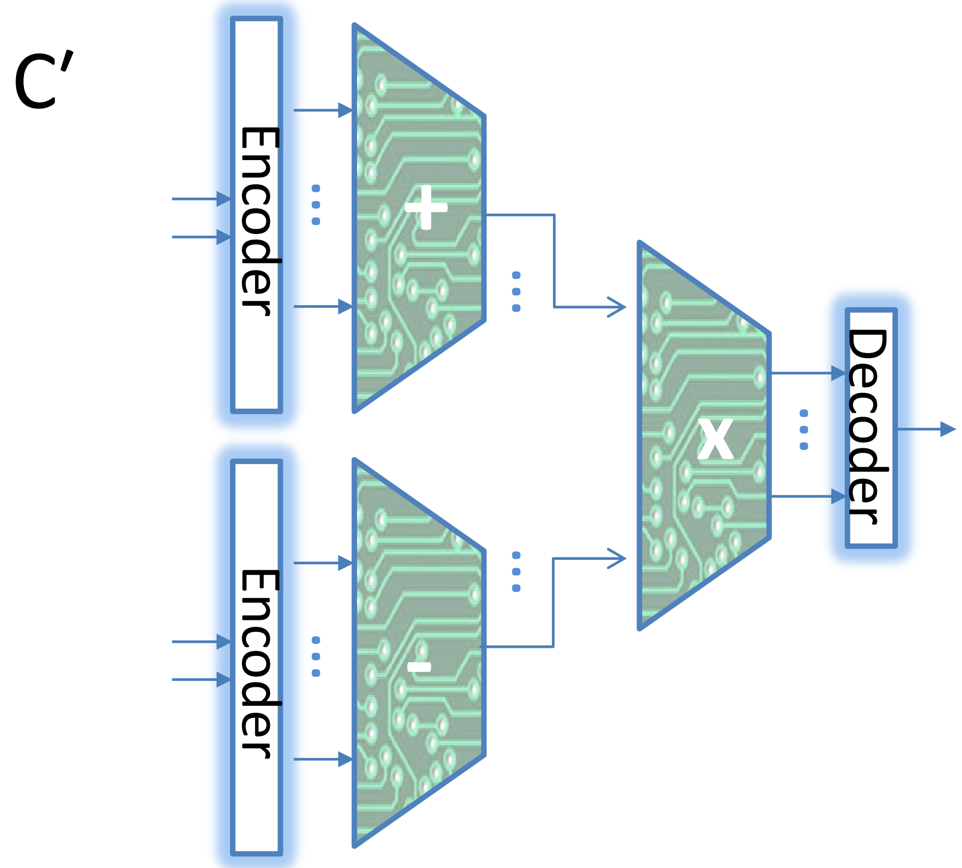
Wire bundle that carries the encoding of w, e.g. k shares of w. <u>Notation:</u> $(w_1,\ldots,w_k)=[w]_d$

PSS is Secure Against AC^0 Leakages

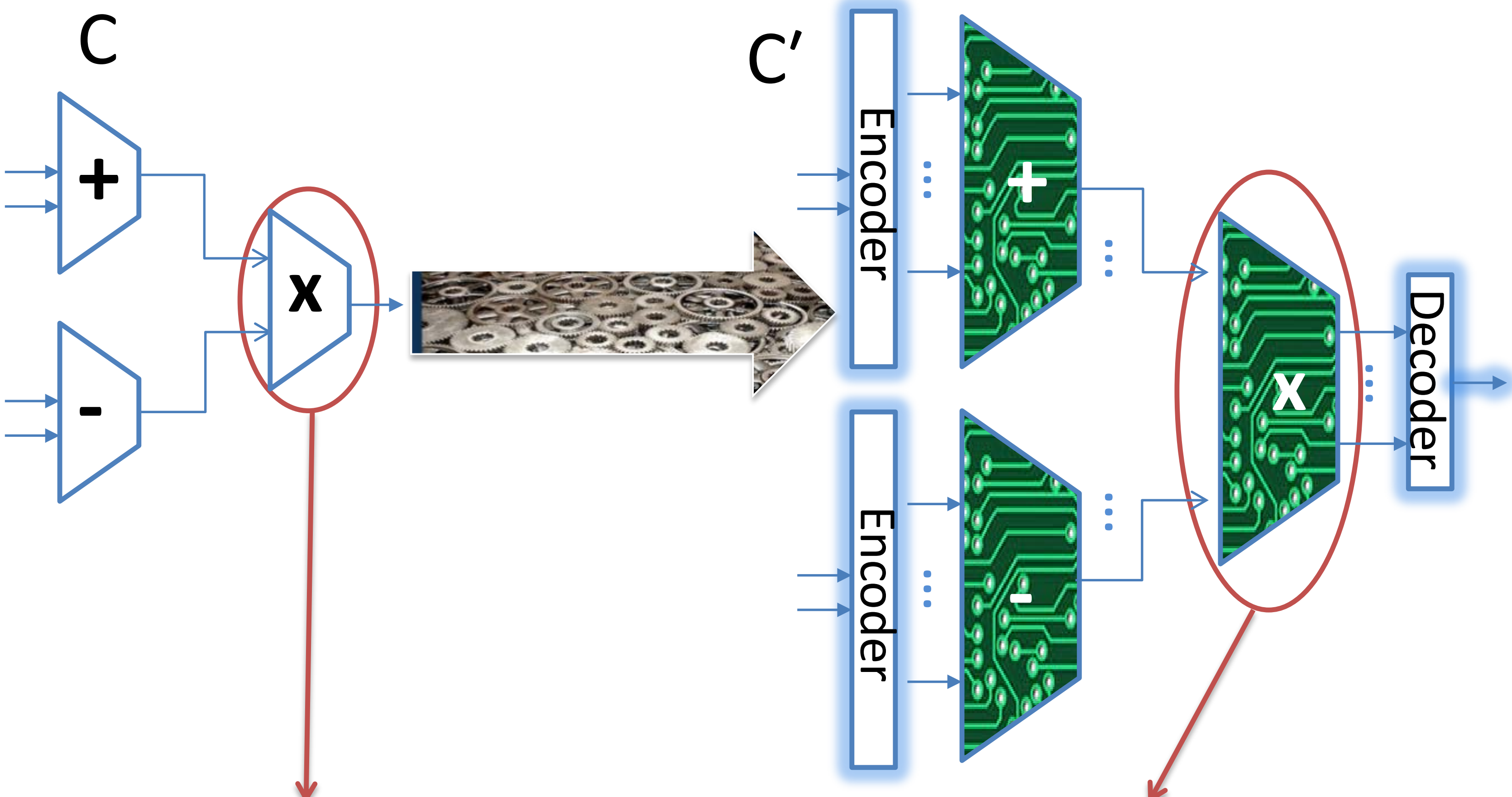
A function is in AC^0 if it can be computed by a poly-size $O(1)$ depth Boolean circuit with unbounded fan-in AND, OR (and NOT) gates.

- PSS Encoding is AC^0 indistinguishable, i.e. Inner product hard to compute in AC^0.

The Compiler:
From Gates to Gadgets



- Every gate is replaced by a gadget operating on encoded PSS bundles.



Gates

Gadgets: built from normal gates and opaque gates and operate on encodings.

Opaque Gates

[G89,GoldOstr95]…Leak-free processor: oblivious RAM

[MR04], [DP08], [GKR08], [DF12]…Leak-free memory: “only computation leaks”, one-time programs

[FRRTV10],… Opaque Gates

[GR12],[R13]… Ciphertext banks

Opaque Gates: simple gates that sample from a fixed distribution

e.g.: **securely draw strings with inner product 0**.

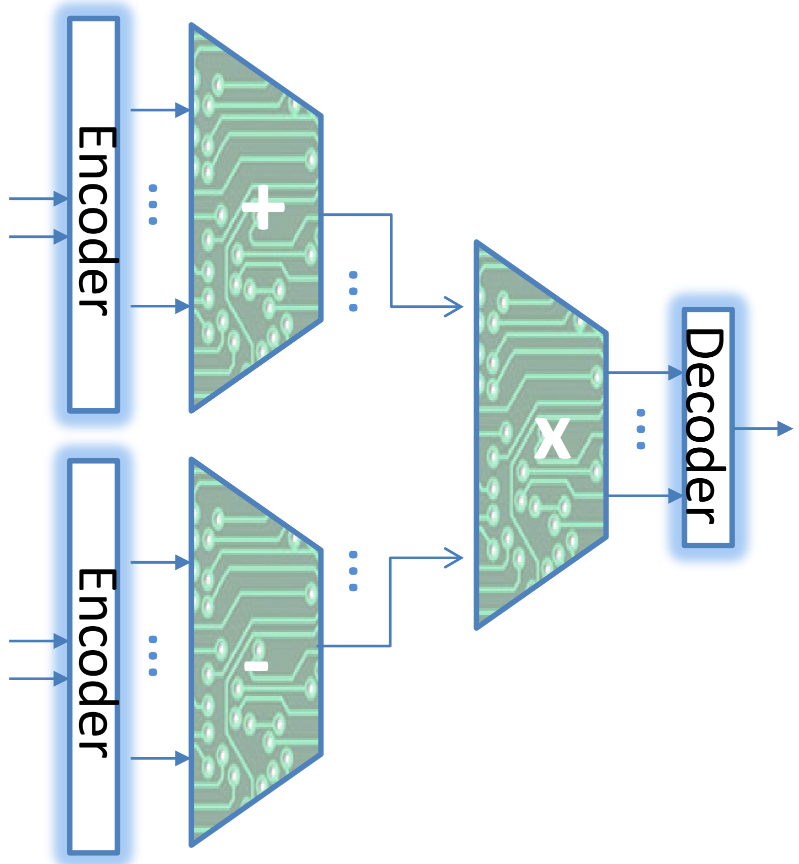
✓ Stateless: No secrets are stored

✓ Small and simple

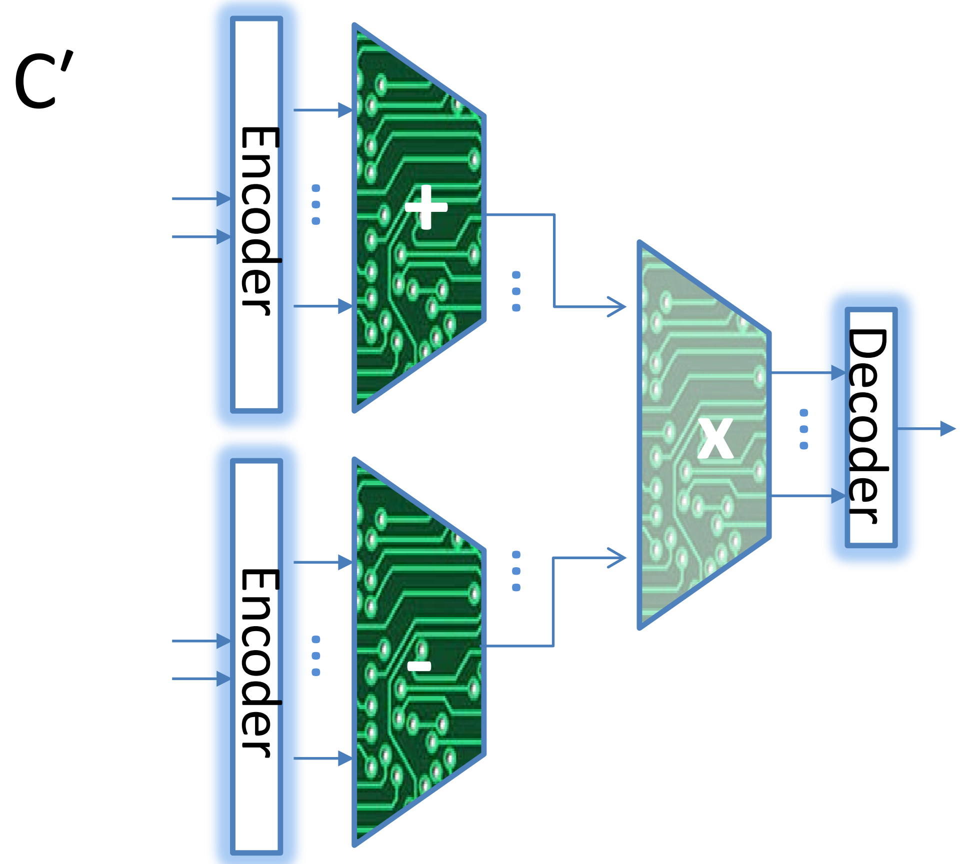
✓ Computation independent: No inputs, so can be pre-computed

The Compiler:
Addition & Subtraction Gadgets

C′



The Compiler:
Addition & Subtraction Gadgets



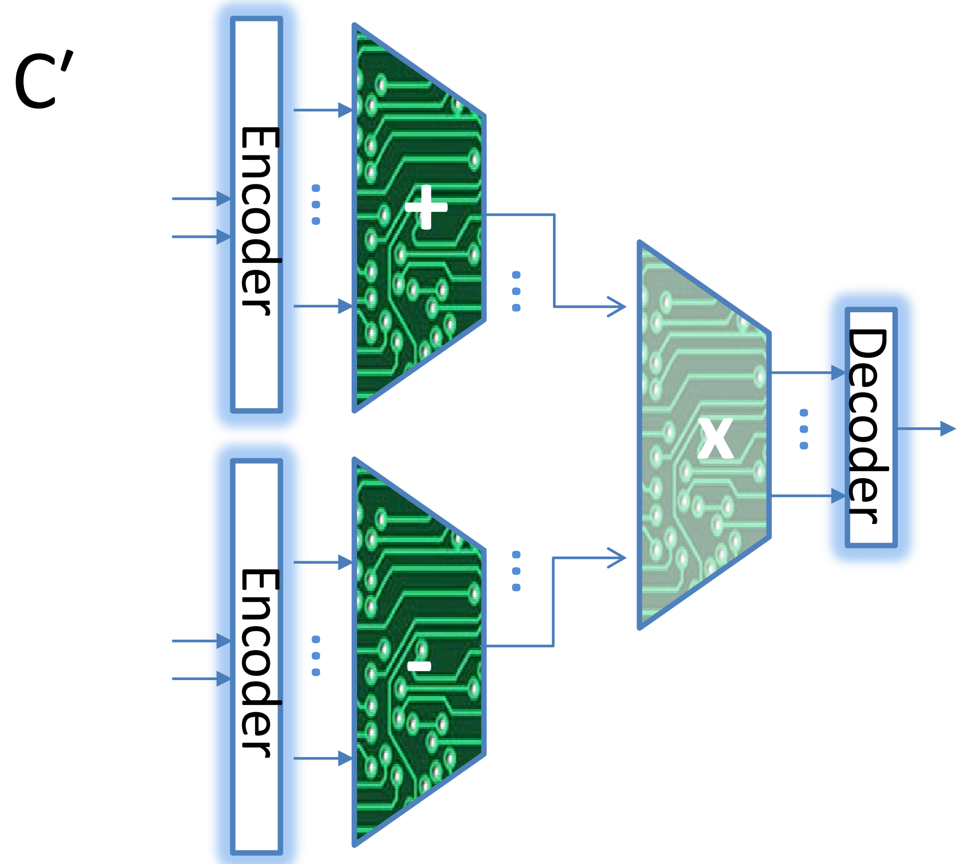
The Compiler: Addition & Subtraction Gadgets

Goal : $c=a+b \Rightarrow [a+b]_d \leftarrow [a]_d + [b]_d$

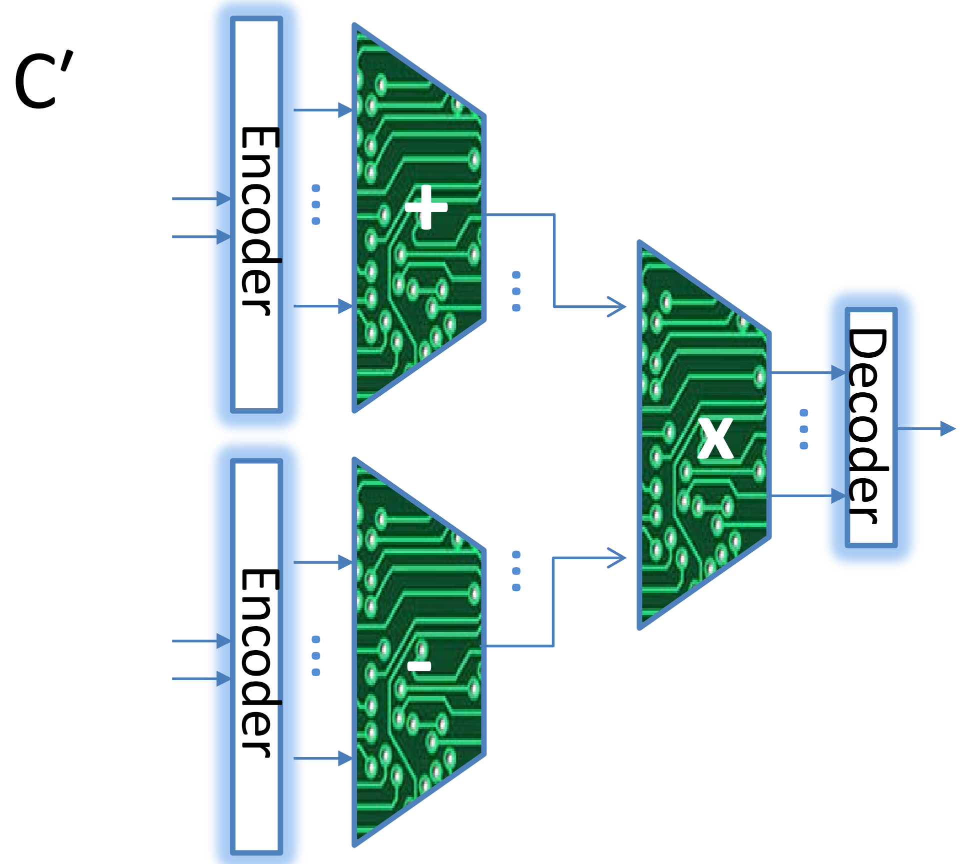
$[0]_d \leftarrow$ Opaque gate

1. $[a+b]_d = [a]_d+[b]_d+[0]_d$ OR $[a-b]_d = [a]_d-[b]_d+[0]_d$

The Compiler:
Multiplication Gadgets



The Compiler:
Multiplication Gadgets



The Compiler: Multiplication Gadgets

Goal: $c = ab \Rightarrow [ab]_d \leftarrow [a]_d[b]_d$

$[r]_d, [r]_{2d} \leftarrow$ Opaque gate

1. $[ab]_{2d} = [a]_d[b]_d$
2. $[ab + r]_{2d} = [ab]_{2d} + [r]_{2d}$
3. $(\mathbf{ab+r}) \leftarrow$ Decode$([ab+r]_{2d})$
4. $[ab+r]_d \leftarrow$ Encode $(\mathbf{ab+r})$
5. $[ab]_d = [ab+r]_d - [r]_d$

Permutation gadget follows in the same fashion.

Compiler: High-Level

- Circuit topology is preserved.
- Every wire is encoded yielding a wire bundle; Inputs are encoded; outputs are decoded.
- PSS Encoding is AC^0 indistinguishable
- Every gate is converted into a gadget operating on encodings.

Security of the Compiled Circuit

Prove security via ‘*shallow*’ Reconstructors per gadget (technique introduced in [FRRTV10])

- Reconstructor: on input the inputs and the outputs of a gadget is able to simulate its internals in a way that looks indistinguishable for leakages from AC^0.

Conclusion

Three efficient circuit compilers ….

- ✓ compile any circuit
- ✓ ‘Local’ Wire-Probing
- ✓ ‘Local’ OC Leakage
- ✓ ‘Global’ Computational weak Leakage

Question

Connection to Obfuscation

Thank you!

Optimally Efficient Multi-Party Fair Exchange and Fair Secure Multi-Party Computation

Handan Kılınç [1]   Alptekin Küpçü [2]

[1]EPFL, Koç University
[2]Koç University

CT-RSA, 2015

Outline

1. Introduction
   - Multi-Party Fair Exchange
   - Definitions
2. Our New Protocols
   - MFE Protocol
     - Resolve Protocols
   - Fair and Secure MPC
3. Conclusion
   - Security and Fairness
   - Comparison with Previous Works
   - Conclusion

Outline

1 Introduction
- Multi-Party Fair Exchange
- Definitions

2 Our New Protocols
- MFE Protocol
  - Resolve Protocols
- Fair and Secure MPC

3 Conclusion
- Security and Fairness
- Comparison with Previous Works
- Conclusion

MFE

Exchange Protocol

Two or more parties exchange their items with the other parties.

Fair Exchange Protocol

The exchange protocol is fair if in the end of

- All parties receive their desired items or,
- None of them receives any item.

MFE

Exchange Protocol

Two or more parties exchange their items with the other parties.

Fair Exchange Protocol

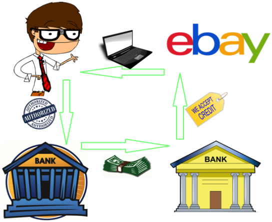
The exchange protocol is fair if in the end of

- All parties receive their desired items or,
- None of them receives any item.

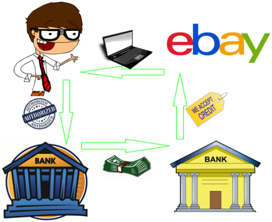
Where is MFE used?

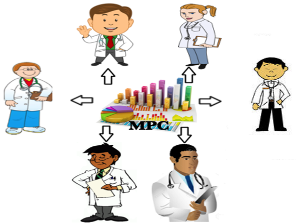
Where is MFE used?

Where is MFE used?

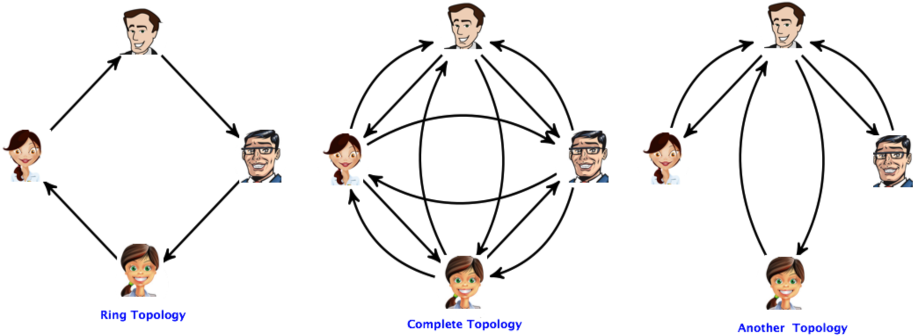


Where is MFE used?





MFE Topologies



Optimistic MFE

- ☹Fairness is not possible without trusted third party (TTP).
- ☹There is a lack of TTP. So the efficiency is important.

Optimistic MFE

- ☹Fairness is not possible without trusted third party (TTP).
- ☹There is a lack of TTP. So the efficiency is important.

Optimistic MFE

- ☹Fairness is not possible without trusted third party (TTP)
- ☹There is a lack of TTP. So efficiency is important.

Optimistic MFE ☺

In an *optimistic* protocol, the TTP is involved in the protocol *only* when there is a malicious behavior.

Optimistic MFE

- ☹Fairness is not possible without trusted third party (TTP)
- ☹There is a lack of TTP. So efficiency is important.

Optimistic MFE ☺

In an *optimistic* protocol, the TTP is involved in the protocol *only* when there is a malicious behavior.

Outline

1. Introduction
   - Multi-Party Fair Exchange
   - Definitions
2. Our New Protocols
   - MFE Protocol
     - Resolve Protocols
   - Fair and Secure MPC
3. Conclusion
   - Security and Fairness
   - Comparison with Previous Works
   - Conclusion

Multi-Party Computation

MPC

A group of parties ($P_1, P_2, ..., P_n$) with their private inputs w_i desires to compute a function ϕ.

- This computation is secure when the parties do not learn anything beyond what is revealed by the output of the computation.
- This computation is fair if either all of the parties learn their corresponding output in the end of computation, or none of them learns.

Multi-Party Computation

MPC

A group of parties $(P_1, P_2, ..., P_n)$ with their private inputs w_i desires to compute a function ϕ.

- This computation is secure when the parties do not learn anything beyond what is revealed by the output of the computation.
- This computation is fair if either all of the parties learn their corresponding output in the end of computation, or none of them learns.

Multi-Party Computation

MPC

A group of parties $(P_1, P_2, ..., P_n)$ with their private inputs w_i desires to compute a function ϕ.

- This computation is secure when the parties do not learn anything beyond what is revealed by the output of the computation.
- This computation is fair if either all of the parties learn their corresponding output in the end of computation, or none of them learns.

MFE is MPC

Multi-party fair exchange is multi-party computation.

- Each party P_i has item f_i.
- They need the compute the functionality ϕ where

$$\phi(f_1, f_2, ..., f_n) = (\phi_1, \phi_2, ..., \phi_n)$$

MFE id MPC

- For the complete topology:

$$\phi_i(f_1, ..., f_n) = (f_1, ..., f_{i-1}, f_{i+1}, ..., f_n)$$

- For the ring topology:
  if $i = 1$

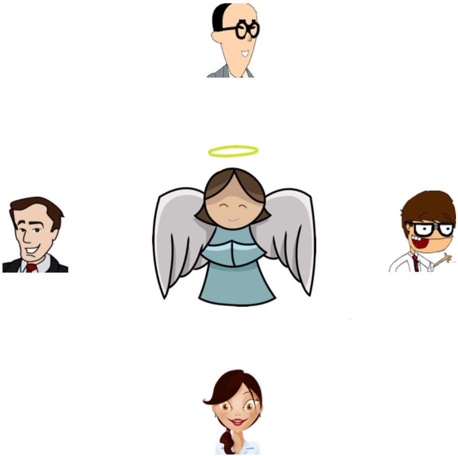
$$\phi_i(f_1, ..., f_n) = f_n$$

  else

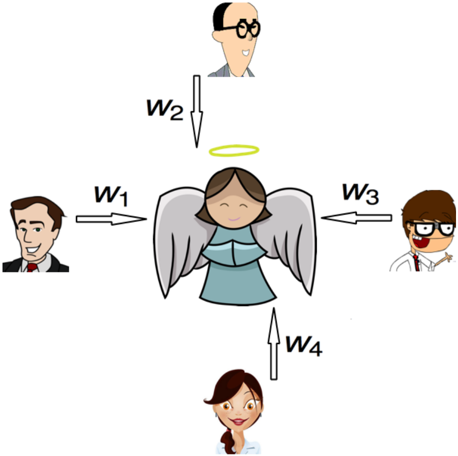
$$\phi_i(f_1, ..., f_n) = f_{i-1}$$

Ideal World for Fair and Secure MPC

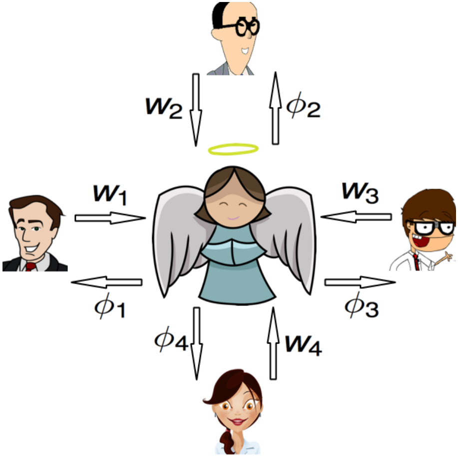
Ideal World for Fair and Secure MPC



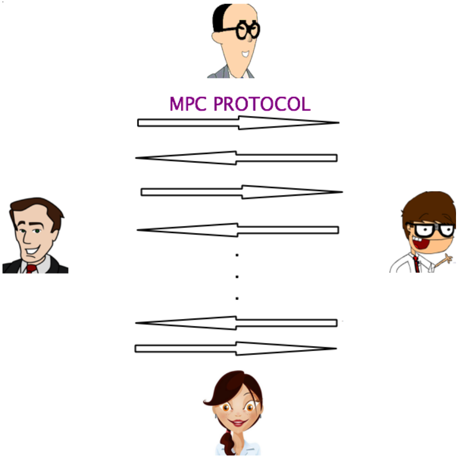
Ideal World for Fair and Secure MPC



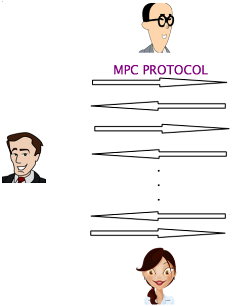
Ideal World for Fair and Secure MPC



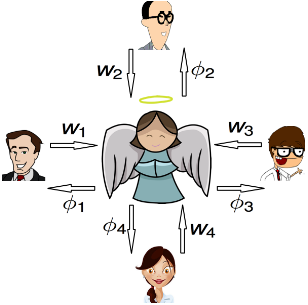
Real World for MPC



Secure and Fair MPC



$\approx$



Outline

1. Introduction
   - Multi-Party Fair Exchange
   - Definitions
2. Our New Protocols
   - MFE Protocol
     - Resolve Protocols
   - Fair and Secure MPC
3. Conclusion
   - Security and Fairness
   - Comparison with Previous Works
   - Conclusion

Overview of MFE protocol

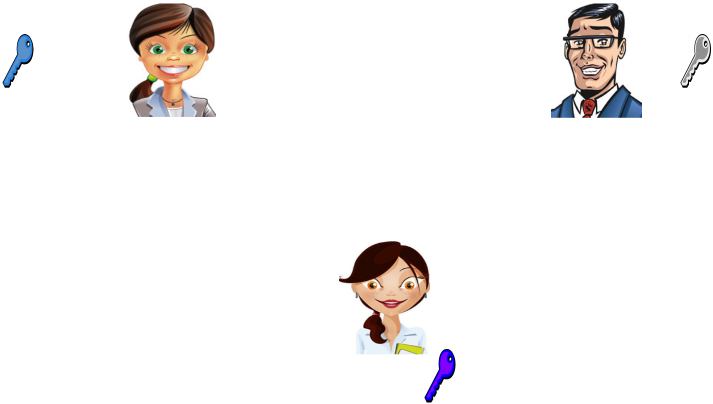
The parties are $P_1, P_2, ..., P_n$ and each party P_i has item f_i. They want the items of all parties (complete topology).
The TTP and his public key pk is known by all parties.

- Phase 1: Setup
- Phase 2: Encrypted Item Exchange
- Phase 3: Decryption Share Exchange

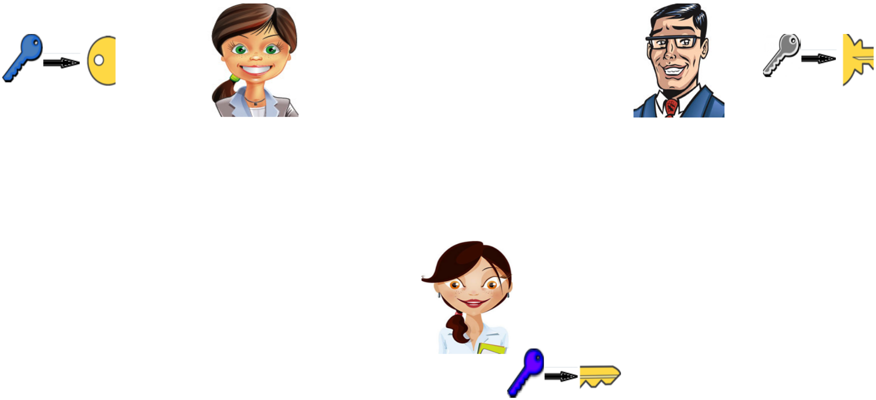
Phase 1: Setup Phase

They agree on two timeouts t_1 and t_2 and know TTP's public key

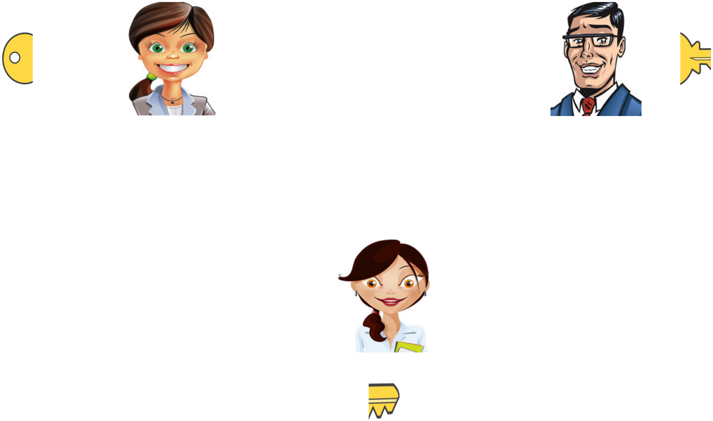
Phase 1: Setup Phase



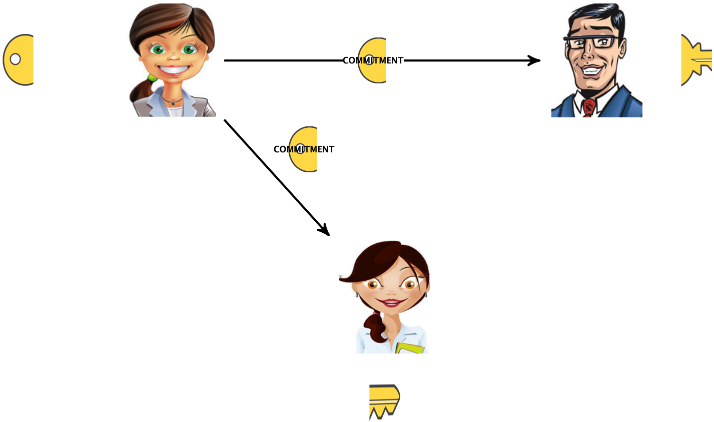
Phase 1: Setup Phase



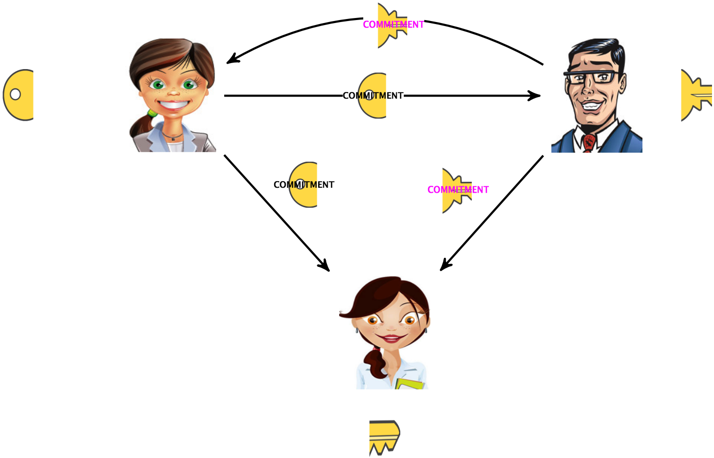
Phase 1: Setup Phase



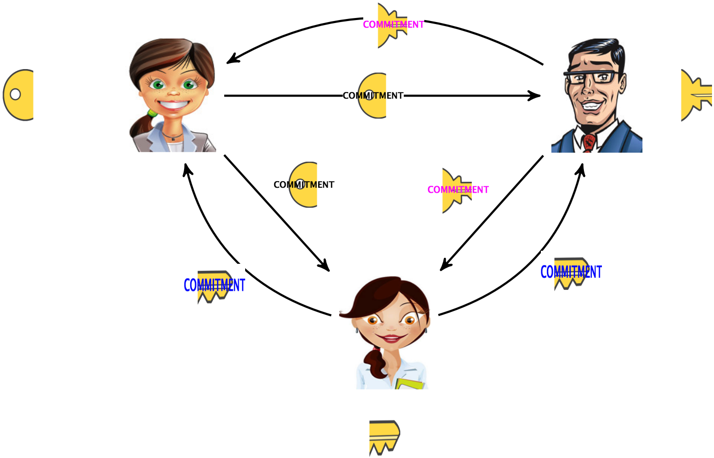
Phase 1: Setup Phase



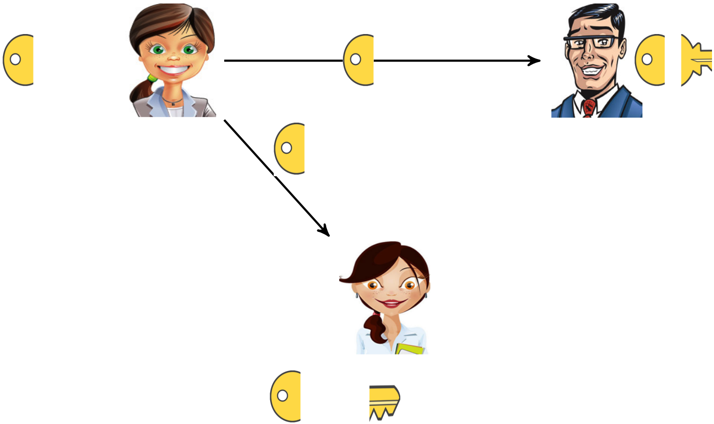
Phase 1: Setup Phase



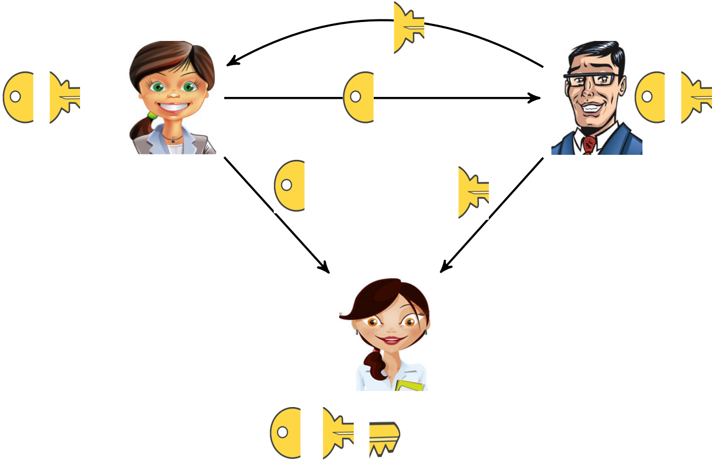
Phase 1: Setup Phase



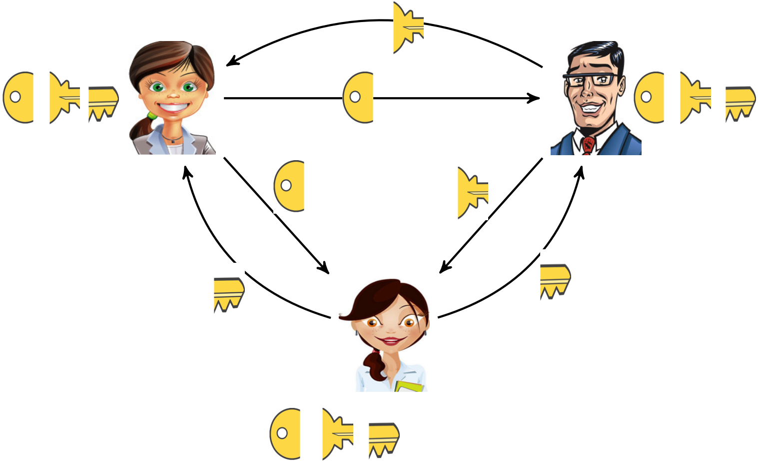
Phase 1: Setup Phase



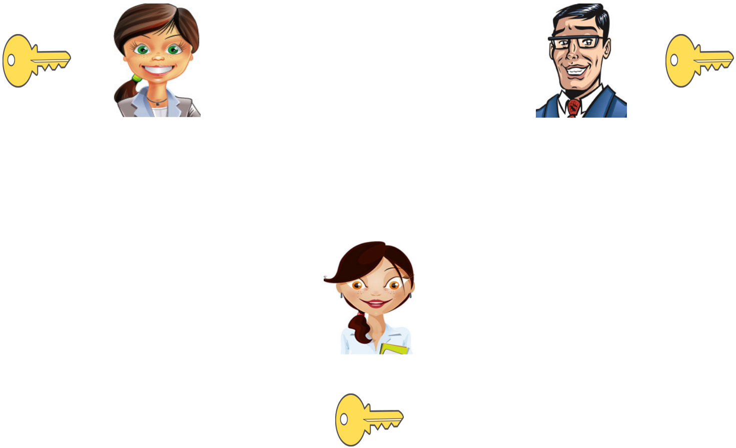
Phase 1: Setup Phase



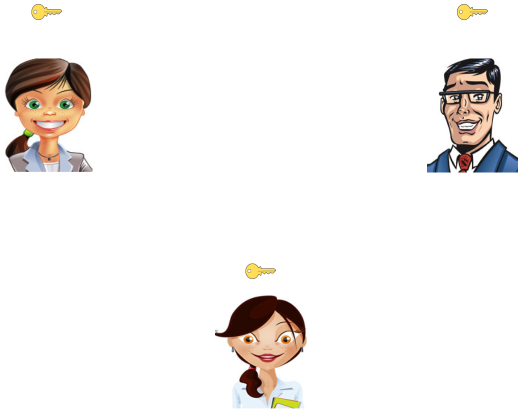
Phase 1: Setup Phase



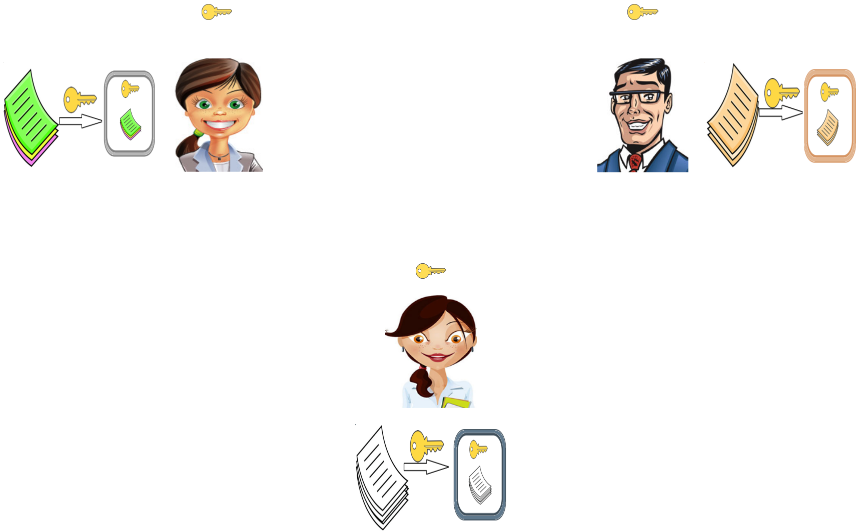
Phase 1: Setup Phase



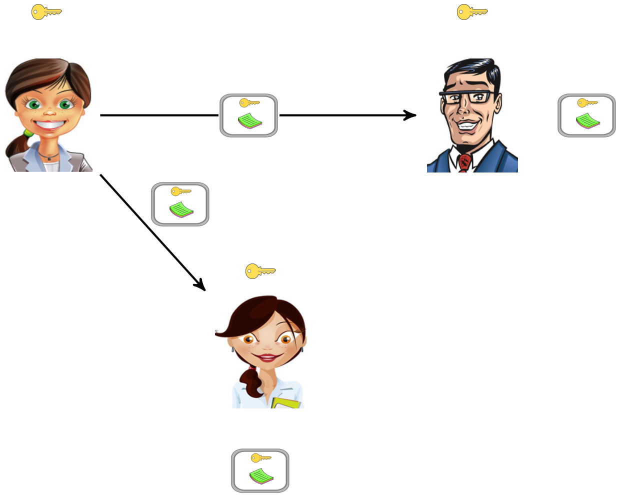
Phase 2: Verifiable Encryption of Items



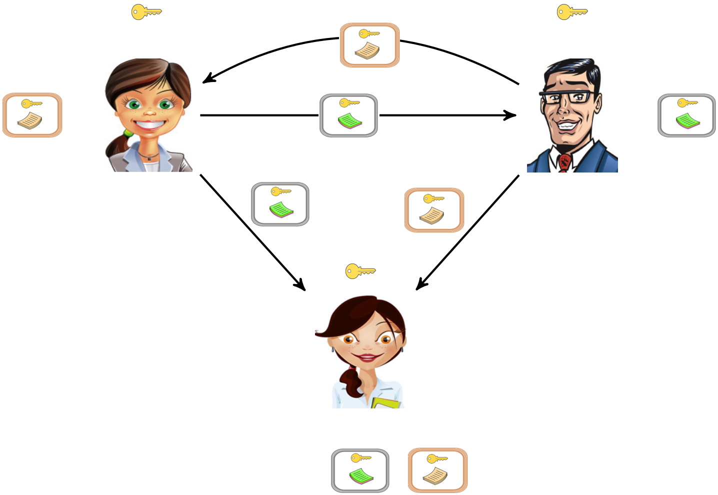
Phase 2: Verifiable Encryption of Items



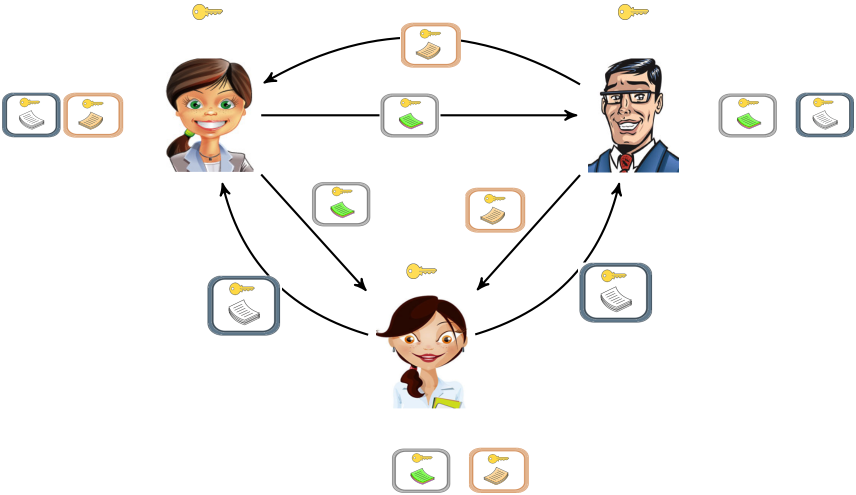
Phase 2: Verifiable Encryption of Items



Phase 2: Verifiable Encryption of Items



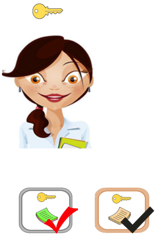
Phase 2: Verifiable Encryption of Items



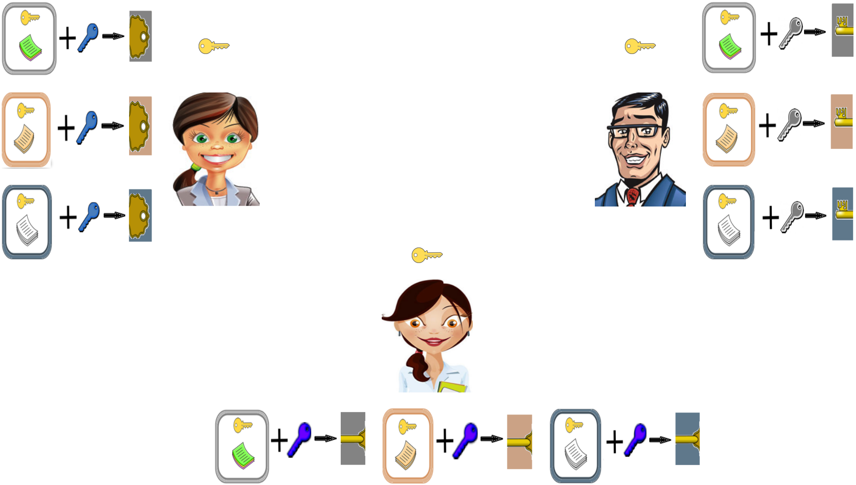
Phase 2: Verifiable Encryption of Items

Phase 2: Verifiable Encryption of Items

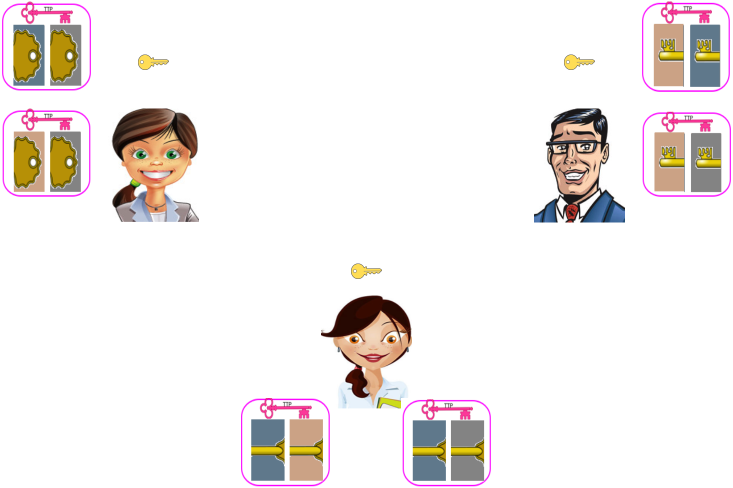
If any party does not receive verifiable encryption, (s)he aborts.



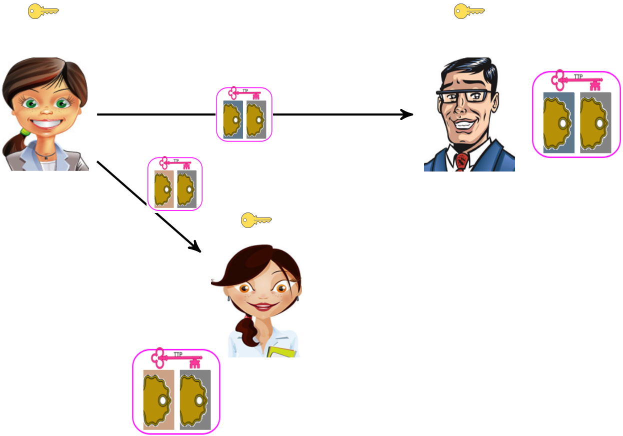
Phase 2: Decryption Share Encryption



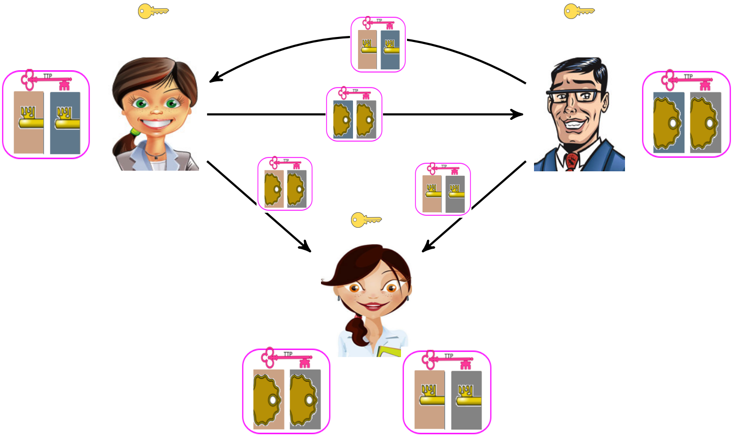
Phase 2: Decryption Share Encryption



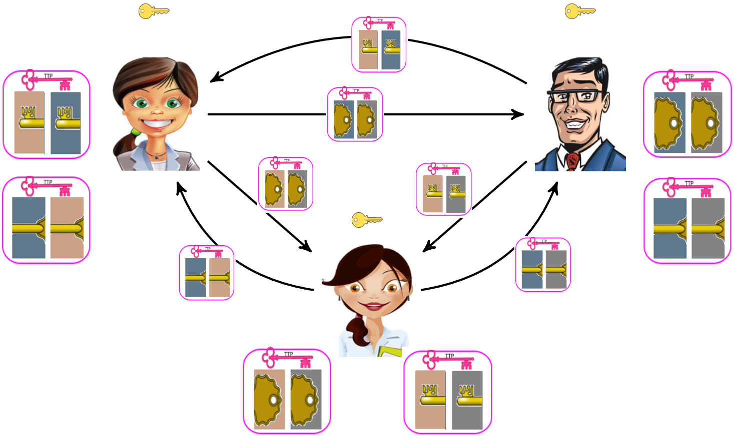
Phase 2: Decryption Share Encryption



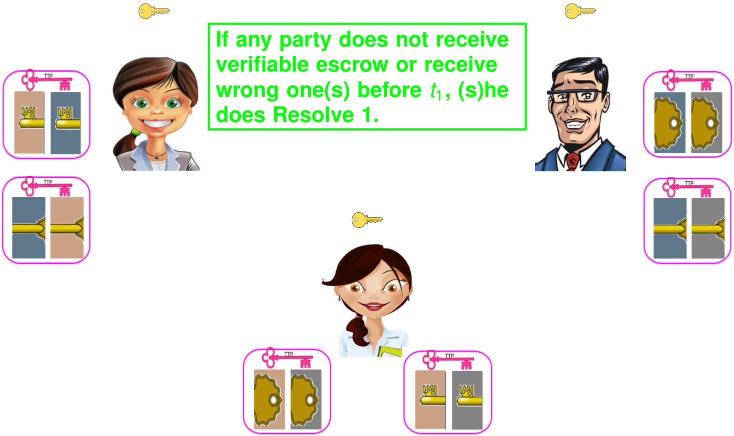
Phase 2: Decryption Share Encryption



Phase 2: Decryption Share Encryption

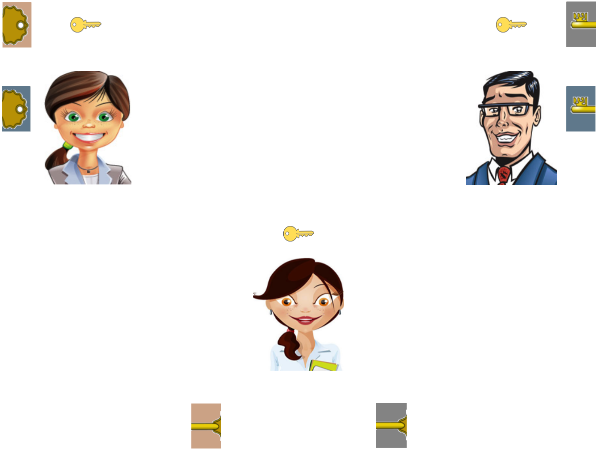


Phase 2: Decryption Share Encryption

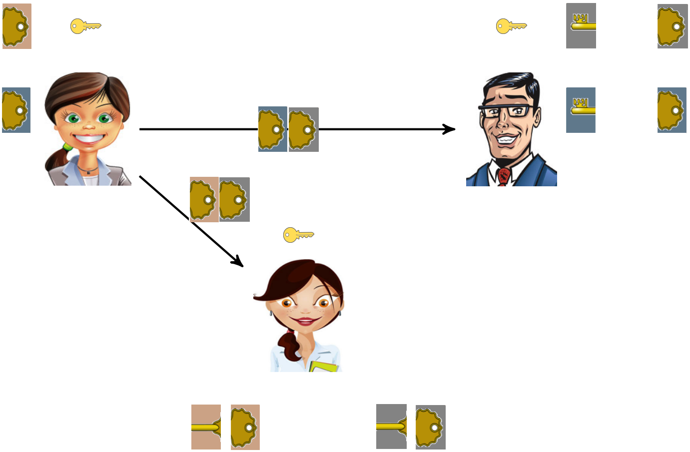


If any party does not receive verifiable escrow or receive wrong one(s) before t_1, (s)he does Resolve 1.

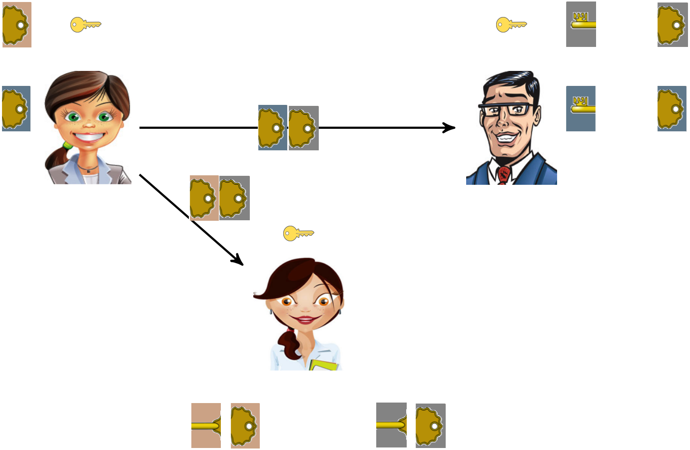
Phase 3: Decryption Share Exchange



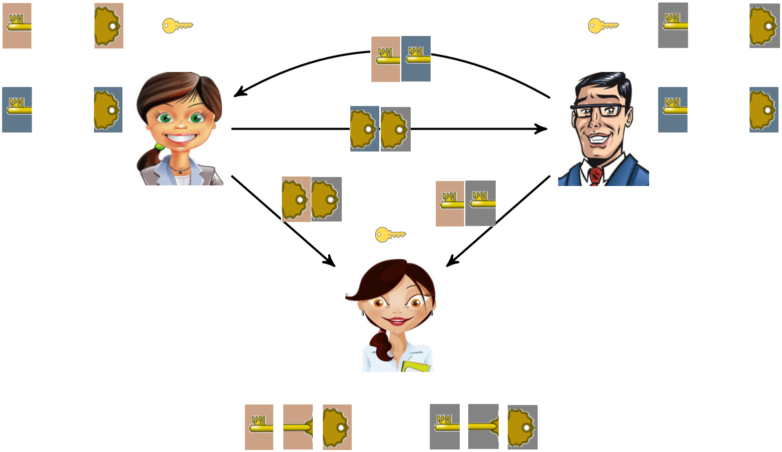
Phase 3: Decryption Share Exchange



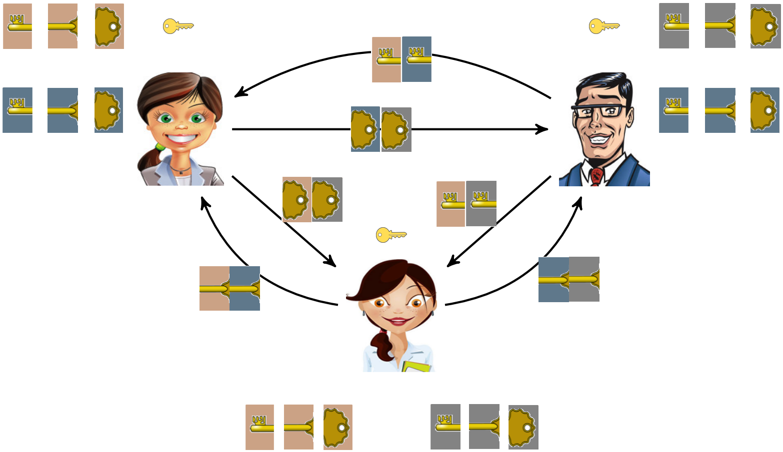
Phase 3: Decryption Share Exchange



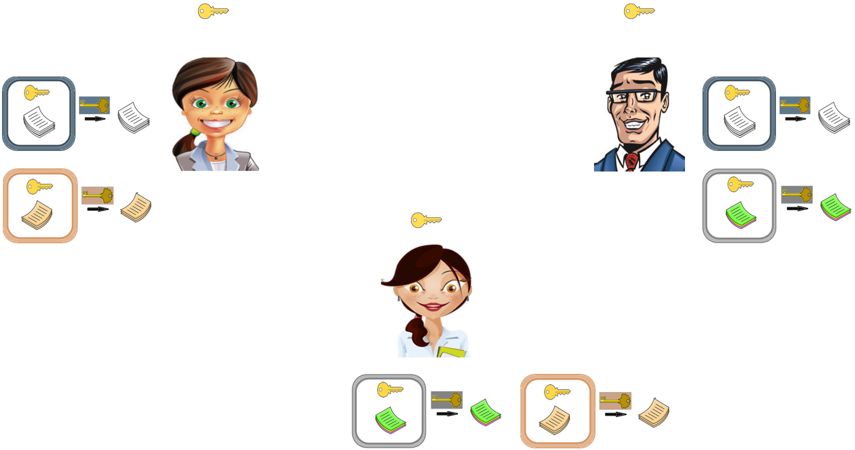
Phase 3: Decryption Share Exchange



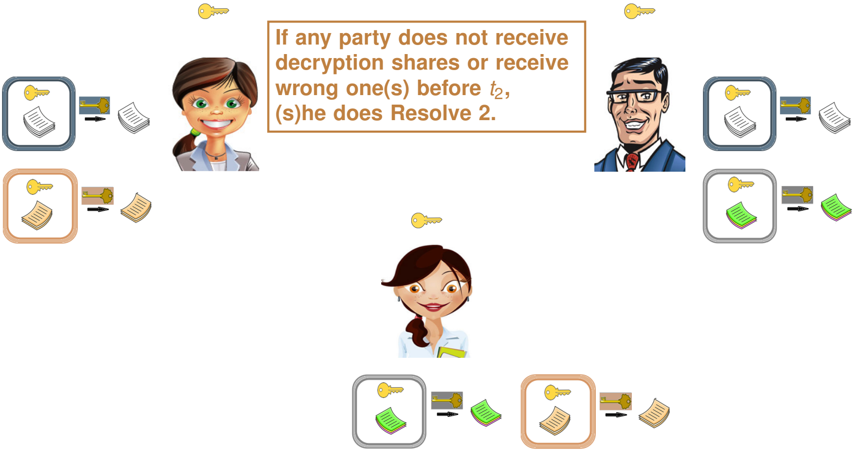
Phase 3: Decryption Share Exchange



Phase 3: Decryption Share Exchange



Phase 3: Decryption Share Exchange



If any party does not receive decryption shares or receive wrong one(s) before t_2, (s)he does Resolve 2.

Resolve 1

- Parties do *not* learn any decryption shares here.
- They can just complain about other parties to the TTP.
- The TTP creates a fresh *complaintList* for the protocol with parameters id, t_1, t_2.

Resolve 2

- The party P_i, who comes for Resolve 2 **between $\mathbf{t_1}$ and $\mathbf{t_2}$**, gives all verifiable escrows that he has already received from the other parties and his own verifiable escrow to the TTP.
- The TTP uses these verifiable escrows to save the decryption shares required to solve the complaints in the *complaintList*.
  - If the *complaintList* is not empty in the end, P_i comes after t_2 for **Resolve 3**.
  - Otherwise, TTP decrypts the verifiable escrow and gives decryption shares.

Resolve 3

- If the *complaintList* still has parties, even after t_2, the TTP answers each resolving party saying that the protocol is **aborted**, which means nobody is able to learn any item.
- If the *complaintList* is *empty*, the TTP decrypts any verifiable escrow that is given to him.

Outline

1. Introduction
   - Multi-Party Fair Exchange
   - Definitions
2. Our New Protocols
   - MFE Protocol
     - Resolve Protocols
   - Fair and Secure MPC
3. Conclusion
   - Security and Fairness
   - Comparison with Previous Works
   - Conclusion

Making SMPC Fair with MFE

SMPC

Parties are able to compute the following function in a secure way by using SMPC protocol.

$$\phi(w_1, ..., w_n) = (\phi_1(w_1, ..., w_n), ..., \phi_n(w_1, ..., w_n))$$

Making SMPC Fair with MFE

Fair SMPC

- Change input of the each P_i as $z_i = (w_i, x_i)$.
- Compute the following functionality with SMPC.

$$\psi_i(z_1, z_2, ..., z_n) = (E_i(\phi_i(w_1, ..., w_n)), \{g^{x_j}\}_{1 \leq j \leq n})$$

where

$$E_i(\phi_i(w_1, ..., w_n)) = (g^{r_i}, \phi_i h^{r_i})$$

- Execute Phase 3 of MFE protocol.

Making SMPC Fair with MFE

Fair SMPC

- Change input of the each P_i as $z_i = (w_i, x_i)$.
- Compute the following functionality with SMPC.

$$\psi_i(z_1, z_2, ..., z_n) = (E_i(\phi_i(w_1, ..., w_n)), \{g^{x_j}\}_{1 \leq j \leq n})$$

where

$$E_i(\phi_i(w_1, ..., w_n)) = (g^{r_i}, \phi_i h^{r_i})$$

- Execute Phase 3 of MFE protocol.

Making SMPC Fair with MFE

Fair SMPC

- Change input of the each P_i as $z_i = (w_i, x_i)$.
- Compute the following functionality with SMPC.

$$\psi_i(z_1, z_2, ..., z_n) = (E_i(\phi_i(w_1, ..., w_n)), \{g^{x_j}\}_{1 \leq j \leq n})$$

where

$$E_i(\phi_i(w_1, ..., w_n)) = (g^{r_i}, \phi_i h^{r_i})$$

- Execute Phase 3 of MFE protocol.

Making SMPC Fair with MFE

Fair SMPC

- Change input of the each P_i as $z_i = (w_i, x_i)$.
- Compute the following functionality with SMPC.

$$\psi_i(z_1, z_2, ..., z_n) = (E_i(\phi_i(w_1, ..., w_n)), \{g^{x_j}\}_{1 \leq j \leq n})$$

where

$$E_i(\phi_i(w_1, ..., w_n)) = (g^{r_i}, \phi_i h^{r_i})$$

- Execute Phase 3 of MFE protocol.

Outline

1 Introduction
- Multi-Party Fair Exchange
- Definitions

2 Our New Protocols
- MFE Protocol
  - Resolve Protocols
- Fair and Secure MPC

3 Conclusion
- Security and Fairness
- Comparison with Previous Works
- Conclusion

Why MFE is fair?

- Parties do not learn anything without any missing decryption share.
  ⇒ All parties depend each other. So even though $n-1$ malicious party exist, they can not exclude an honest party.
- If an honest party does not receive verifiable escrow, (s)he does not continue.
  ⇒ This obliges malicious party to send his verifiable escrow to the honest party, otherwise malicious one cannot learn anything.
- TTP does not decrypt verifiable escrow and send any decryption share until it is sure that he has all missing verifiable escrows.
  ⇒ Resolve protocols preserve fairness.

Why MFE is fair?

- Parties do not learn anything without any missing decryption share.
  $\Rightarrow$ All parties depend each other. So even though $n-1$ malicious party exist, they can not exclude an honest party.
- If an honest party does not receive verifiable escrow, (s)he does not continue.
  $\Rightarrow$ This obliges malicious party to send his verifiable escrow to the honest party, otherwise malicious one cannot learn anything.
- TTP does not decrypt verifiable escrow and send any decryption share until it is sure that he has all missing verifiable escrows.
  $\Rightarrow$ Resolve protocols preserve fairness.

Why MFE is fair?

- Parties do not learn anything without any missing decryption share.
  ⇒ All parties depend each other. So even though $n-1$ malicious party exist, they can not exclude an honest party.
- If an honest party does not receive verifiable escrow, (s)he does not continue.
  ⇒ This obliges malicious party to send his verifiable escrow to the honest party, otherwise malicious one cannot learn anything.
- TTP does not decrypt verifiable escrow and send any decryption share until it is sure that he has all missing verifiable escrows.
  ⇒ Resolve protocols preserve fairness.

Privacy in MFE and MPC

Privacy

The **privacy against the TTP is preserved.** He just learns some decryption shares, but he cannot decrypt the encryption of exchanged items, since he never gets the encrypted items.

Outline

1. Introduction
   - Multi-Party Fair Exchange
   - Definitions
2. Our New Protocols
   - MFE Protocol
     - Resolve Protocols
   - Fair and Secure MPC
3. Conclusion
   - Security and Fairness
   - Comparison with Previous Works
   - Conclusion

Previous works in Complete Topology

	Solution	Topology	Round Complexity	Number of Messages	Broad-cast
Garay & MacKenzie	MPCS	Complete	$O(n^2)$	$O(n^3)$	Yes
Baum & Waidner	MPCS	Complete	$O(tn)$	$O(tn^2)$	Yes
Mukhamedov & Ryan	MPCS	Complete	$O(n)$	$O(n^3)$	Yes
Mauw et al.	MPCS	Complete	$O(n)$	$O(n^2)$ ✓	Yes
Asokan et al.	MFE ✓	Any ✓	$O(1)$ ✓	$O(n^3)$	Yes
Ours	MFE ✓	Any ✓	$O(1)$ ✓	$O(n^2)$ ✓	No ✓

Previous works in Ring Topology

	Number Messages	**All or None**	**TTP-Party Dependency**	**TTP Privacy**
Bao et al.	$O(n)$	No	Yes	Not Private
González & Markowitch	$O(n^2)$	No	Yes	Not Private
Liu & Hu	$O(n)$	No	Yes	Not Private
Ours	$O(n^2)$	Yes ✓	No ✓	Private ✓

Previous works in fair SMPC

	Technique	**TTP**	**Number of Rounds**	**Proof Technique**
Garay et al.	Gradual Release	No	$O(\lambda)$	NFS
Bentov & Kumaresan	Bitcoin	Yes	Constant ✓	NFS
Andrychowicz et al.	Bitcoin	Yes	Constant ✓	NFS
Ours	MFE	Yes	Constant ✓	FS ✓

Outline

1. Introduction
   - Multi-Party Fair Exchange
   - Definitions
2. Our New Protocols
   - MFE Protocol
     - Resolve Protocols
   - Fair and Secure MPC
3. Conclusion
   - Security and Fairness
   - Comparison with Previous Works
   - Conclusion

Our Contributions

MFE

- ✓ We design a MFE protocol requires only $\mathbf{O}(\mathbf{n^2})$ messages and **constant** number of rounds for n parties.
- ✓ Our MFE **optimally** (in complete topology) guarantees fairness (for honest parties) even when $n-1$ out of n parties are malicious and colluding.
- ✓ We show how to employ our MFE protocol for **any exchange topology**, with the performance improving as the topology gets sparser.
- ✓ We formulate MFE as a secure multi-party computation protocol. We then **prove** security and fairness **via ideal-real world simulation** [9].

Our Contributions

MFE

- ✓ We design a MFE protocol requires only $\mathbf{O}(\mathbf{n^2})$ messages and **constant** number of rounds for n parties.
- ✓ Our MFE **optimally** (in complete topology) guarantees fairness (for honest parties) even when $n-1$ out of n parties are malicious and colluding.
- ✓ We show how to employ our MFE protocol for **any exchange topology**, with the performance improving as the topology gets sparser.
- ✓ We formulate MFE as a secure multi-party computation protocol. We then **prove** security and fairness **via ideal-real world simulation** [9].

Our Contributions

MFE

- ✓ We design a MFE protocol requires only $\mathbf{O}(\mathbf{n^2})$ messages and **constant** number of rounds for n parties.
- ✓ Our MFE **optimally** (in complete topology) guarantees fairness (for honest parties) even when $n-1$ out of n parties are malicious and colluding.
- ✓ We show how to employ our MFE protocol for **any exchange topology**, with the performance improving as the topology gets sparser.
- ✓ We formulate MFE as a secure multi-party computation protocol. We then **prove** security and fairness **via ideal-real world simulation** [9].

Our Contributions

MFE

- ✓ We design a MFE protocol requires only $\mathbf{O}(\mathbf{n^2})$ messages and **constant** number of rounds for n parties.
- ✓ Our MFE **optimally** (in complete topology) guarantees fairness (for honest parties) even when $n-1$ out of n parties are malicious and colluding.
- ✓ We show how to employ our MFE protocol for **any exchange topology**, with the performance improving as the topology gets sparser.
- ✓ We formulate MFE as a secure multi-party computation protocol. We then **prove** security and fairness **via ideal-real world simulation** [9].

Our Contributions

TTP Usage

- ✓ The TTP for fairness in our MFE is in the *optimistic* model The TTP has a very low workload.
- ✓ The TTP does *not* learn any exchanged item, so **privacy against the TTP** is preserved.

Our Contributions

TTP Usage

- ✓ The TTP for fairness in our MFE is in the *optimistic* model The TTP has a very low workload.
- ✓ The TTP does *not* learn any exchanged item, so **privacy against the TTP** is preserved.

Our Contribution

Secure Multi-party Computation

- ✓ Our MFE can be employed **on top of any SMPC protocol** to obtain a *fair* SMPC protocol,
- ✓ We provide an ideal world definition for *fair SMPC*, and prove security and fairness of a SMPC protocol that use our MFE via simulation.

Our Contribution

Secure Multi-party Computation

- ✓ Our MFE can be employed **on top of any SMPC protocol** to obtain a *fair* SMPC protocol,
- ✓ We provide an ideal world definition for *fair SMPC*, and prove security and fairness of a SMPC protocol that use our MFE via simulation.

Authors

Handan Kılınç
PHD student at EPFL
handan.kilinc@epfl.ch

Asst. Prof. Alptekin Küpçü
at Koç University
akupcu@ku.edu.tr

For Further Reading I

- M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek.
  Secure multiparty computations on bitcoin.
  In *IEEE Symposium on Security and Privacy*, 2014.

- N. Asokan, M. Schunter, and M. Waidner.
  Optimistic protocols for multi-party fair exchange, 1996.

- F. Bao, R. H. Deng, K. Q. Nguyen, and V. Varadharajan.
  Multi-party fair exchange with an off-line trusted neutral party.
  In *DEXA Workshop*, 1999.

- B. Baum-Waidner and M. Waidner.
  Round-optimal and abuse free optimistic multi-party contract signing.
  In *ICALP*, 2000.

For Further Reading II

- I. Bentov and R. Kumaresan.
  How to use bitcoin to design fair protocols.
  In *CRYPTO*.
- J. Garay, P. MacKenzie, M. Prabhakaran, and K. Yang.
  Resource fairness and composability of cryptographic protocols.
  In *Theory of Cryptography*. 2006.
- J. A. Garay and P. D. MacKenzie.
  Abuse-free multi-party contract signing.
  In *DISC*, 1999.
- N. González-Deleito and O. Markowitch.
  An optimistic multi-party fair exchange protocol with reduced trust requirements.
  In *ICISC*, 2002.

For Further Reading III

H. Kılınç and A. Küpçü.
Efficiently making secure two-party computation fair.
Cryptology ePrint Archive, Report 2014/896.

Y. Liu and H. Hu.
An improved protocol for optimistic multi-party fair exchange.
In *EMEIT*, 2011.

S. Mauw, S. Radomirovic, and M. T. Dashti.
Minimal message complexity of asynchronous multi-party contract signing.
In *CSF*, 2009.

A. Mukhamedov and M. D. Ryan.
Fair multi-party contract signing using private contract signatures.
Inf. Comput., pages 272–290, 2008.

Phase 1: Setup Phase

All participants agree on the prime p-order subgroup of $\mathcal{Z}_q^*$, where q is a large prime, and a generator g of this subgroup. Then each P_i does

$\underline{P_i}$	**Phase 1**	$\underline{P_j}$
pick x_i		pick x_j
compute $h_i = g^{x_i}$		compute $h_j = g^{x_j}$
$C_i = Com(h_i, r_i)$	$\xrightarrow{C_i}$	$C_j = Com(h_j, r_j)$
	$\xleftarrow{C_j}$	
	$\xrightarrow{h_i}$	
	$\xleftarrow{h_j}$	

Phase 1: Setup Phase

All participants agree on the prime p-order subgroup of $\mathcal{Z}_q^*$, where q is a large prime, and a generator g of this subgroup. Then each P_i does

$\underline{P_i}$	**Phase 1**	$\underline{P_j}$
pick x_i		pick x_j
compute $h_i = g^{x_i}$		compute $h_j = g^{x_j}$
$C_i = Com(h_i, r_i)$	$\xrightarrow{C_i}$	$C_j = Com(h_j, r_j)$
	$\xleftarrow{C_j}$	
	$\xrightarrow{h_i}$	
	$\xleftarrow{h_j}$	

Phase 1: Setup Phase

All participants agree on the prime p-order subgroup of $\mathcal{Z}_q^*$, where q is a large prime, and a generator g of this subgroup. Then each P_i does

$\underline{P_i}$	**Phase 1**	$\underline{P_j}$
pick x_i		pick x_j
compute $h_i = g^{x_i}$		compute $h_j = g^{x_j}$
$C_i = Com(h_i, r_i)$	$\xrightarrow{\quad C_i \quad}$	$C_j = Com(h_j, r_j)$
	$\xleftarrow{\quad C_j \quad}$	
	$\xrightarrow{\quad h_i \quad}$	
	$\xleftarrow{\quad h_j \quad}$	

Phase 2

$\underline{P_i}$	**Phase 2**	$\underline{P_j}$
compute $h = \prod_{k=0}^{n} h_k$		compute $h = \prod_{k=0}^{n} h_k$
pick r_i		pick r_j
$E_i = (a_i, b_i) = (g^{r_i}, f_i h^{r_i})$	$\xrightarrow{VE_i = V(E_i, h; \emptyset)\{(v_i, f_i) \in R_{item}\}}$	$E_j = (a_j, b_j) = (g^{r_j}, f_j h^{r_j})$
	$\xleftarrow{VE_j = V(E_j, h; \emptyset)\{(v_j, f_j) \in R_{item}\}}$	

If VE is not received from at least one of the parties

Abort

Phase 2

$\underline{P_i}$	**Phase 2**	$\underline{P_j}$
compute $h = \prod_{k=0}^{n} h_k$		compute $h = \prod_{k=0}^{n} h_k$
pick r_i		pick r_j
$E_i = (a_i, b_i) = (g^{r_i}, f_i h^{r_i})$	$\xrightarrow{VE_i = V(E_i, h; \emptyset)\{(v_i, f_i) \in R_{item}\}}$ $\xleftarrow{VE_j = V(E_j, h; \emptyset)\{(v_j, f_j) \in R_{item}\}}$	$E_j = (a_j, b_j) = (g^{r_j}, f_j h^{r_j})$

If VE is not received from at least one of the parties

Abort

Phase 2

$\underline{P_i}$	**Phase 2**	$\underline{P_j}$
compute $h = \prod_{k=0}^{n} h_k$		compute $h = \prod_{k=0}^{n} h_k$
pick r_i		pick r_j
$E_i = (a_i, b_i) = (g^{r_i}, f_i h^{r_i})$	$\xrightarrow{VE_i = V(E_i, h; \emptyset)\{(v_i, f_i) \in R_{item}\}}$ $\xleftarrow{VE_j = V(E_j, h; \emptyset)\{(v_j, f_j) \in R_{item}\}}$	$E_j = (a_j, b_j) = (g^{r_j}, f_j h^{r_j})$

If VE is not received from at least one of the parties

Abort

Phase 2

$\underline{P_i}$	**Phase 2**	$\underline{P_j}$
compute $h = \prod_{k=0}^{n} h_k$		compute $h = \prod_{k=0}^{n} h_k$
pick r_i		pick r_j
$E_i = (a_i, b_i) = (g^{r_i}, f_i h^{r_i})$	$\xrightarrow{VE_i = V(E_i, h; \emptyset)\{(v_i, f_i) \in R_{item}\}}$ $\xleftarrow{VE_j = V(E_j, h; \emptyset)\{(v_j, f_j) \in R_{item}\}}$	$E_j = (a_j, b_j) = (g^{r_j}, f_j h^{r_j})$

If VE is not received from at least one of the parties

Abort

Phase 2

$\underline{P_i}$	**Phase 2**	$\underline{P_j}$
compute $h = \prod_{k=0}^{n} h_k$		compute $h = \prod_{k=0}^{n} h_k$
pick r_i		pick r_j
$E_i = (a_i, b_i) = (g^{r_i}, f_i h^{r_i})$	$\xrightarrow{VE_i = V(E_i, h; \emptyset)\{(v_i, f_i) \in R_{item}\}}$ $\xleftarrow{VE_j = V(E_j, h; \emptyset)\{(v_j, f_j) \in R_{item}\}}$	$E_j = (a_j, b_j) = (g^{r_j}, f_j h^{r_j})$

If VE is not received from at least one of the parties

Abort

Phase 3

Note that $\mathbf{a_k} = \mathbf{g^{r_k}}$ (First part of the k^{th} item's encryption).
The relation R_s is $\mathbf{\log_g h_i} = \mathbf{\log_{a_k} a_k^{x_i}}$ for each k.

$\underline{P_i}$	**Phase 3**	$\underline{P_j}$
compute $\{d_k^i = a_k^{x_i}\}_{k=1}^n$	$\xrightarrow{VS_i = V(E_i^t, pk; t_1, t_2, id, P_i)\{(h_i, \{d_k^i\}) \in R_s\}}$	compute $\{d_k^j = a_k^{x_j}\}_{k=1}^n$
$E_i^t = Enc_{pk}(\{d_k^i\}_{k=1}^n)$	$\xleftarrow{VS_j = V(E_j^t, pk; t_1, t_2, id, P_j)\{(h_j, \{d_k^j\}) \in R_s\}}$	$E_j^t = Enc_{pk}(\{d_k^j\}_{k=1}^n)$

If VS is not received from at least one of the parties before time t_1

Resolve 1

$$\xrightarrow{\{d_k^i\} j, PK(h_i, \{a_k\})\{(h_i, \{d_k^i\}) \in R_s\}}$$

$$\xleftarrow{\{d_k^j\}, PK(h_j, \{a_k\})\{(h_j, \{d_k^j\}) \in R_s\}}$$

If d_k^j are not received before t_2

Resolve 2

Phase 3

Note that $\mathbf{a_k} = \mathbf{g^{r_k}}$ (First part of the k^{th} item's encryption).
The relation R_s is $\mathbf{\log_g h_i} = \mathbf{\log_{a_k} a_k^{x_i}}$ for each k.

$\underline{P_i}$	**Phase 3**	$\underline{P_j}$
compute $\{d_k^i = a_k^{x_i}\}_{k=1}^n$	$\xrightarrow{VS_i = V(E_i^t, pk; t_1, t_2, id, P_i)\{(h_i, \{d_k^i\}) \in R_s\}}$	compute $\{d_k^j = a_k^{x_j}\}_{k=1}^n$
$E_i^t = Enc_{pk}(\{d_k^i\}_{k=1}^n)$	$\xleftarrow{VS_j = V(E_j^t, pk; t_1, t_2, id, P_j)\{(h_j, \{d_k^j\}) \in R_s\}}$	$E_j^t = Enc_{pk}(\{d_k^j\}_{k=1}^n)$

If VS is not received from at least one of the parties before time t_1

Resolve 1

$$\xrightarrow{\{d_k^i\} j, PK(h_i, \{a_k\})\{(h_i, \{d_k^i\}) \in R_s\}}$$

$$\xleftarrow{\{d_k^j\}, PK(h_j, \{a_k\})\{(h_j, \{d_k^j\}) \in R_s\}}$$

If d_k^j are not received before t_2

Resolve 2

Phase 3

Note that $\mathbf{a_k} = \mathbf{g^{r_k}}$ (First part of the k^{th} item's encryption).
The relation R_s is $\mathbf{log_g h_i} = \mathbf{log_{a_k} a_k^{x_i}}$ for each k.

$\underline{P_i}$ **Phase 3** $\underline{P_j}$

compute $\{d_k^i = a_k^{x_i}\}_{k=1}^n$ $\xrightarrow{VS_i = V(E_i^t, pk; t_1, t_2, id, P_i)\{(h_i, \{d_k^i\}) \in R_s\}}$ compute $\{d_k^j = a_k^{x_j}\}_{k=1}^n$

$E_i^t = Enc_{pk}(\{d_k^i\}_{k=1}^n)$ $\xleftarrow{VS_j = V(E_j^t, pk; t_1, t_2, id, P_j)\{(h_j, \{d_k^j\}) \in R_s\}}$ $E_j^t = Enc_{pk}(\{d_k^j\}_{k=1}^n)$

If VS is not received from at least one of the parties before time t_1

Resolve 1

$\xrightarrow{\{d_k^i\}j, PK(h_i, \{a_k\})\{(h_i, \{d_k^i\}) \in R_s\}}$

$\xleftarrow{\{d_k^j\}, PK(h_j, \{a_k\})\{(h_j, \{d_k^j\}) \in R_s\}}$

If d_k^i are not received before t_2

Resolve 2

Phase 3

Note that $\mathbf{a_k} = \mathbf{g^{r_k}}$ (First part of the k^{th} item's encryption).
The relation R_s is $\mathbf{\log_g h_i} = \mathbf{\log_{a_k} a_k^{x_i}}$ for each k.

P_i | **Phase 3** | P_j

compute $\{d_k^i = a_k^{x_i}\}_{k=1}^n$ | $\xrightarrow{VS_i = V(E_i^t, pk; t_1, t_2, id, P_i)\{(h_i, \{d_k^i\}) \in R_s\}}$ | compute $\{d_k^j = a_k^{x_j}\}_{k=1}^n$

$E_i^t = Enc_{pk}(\{d_k^i\}_{k=1}^n)$ | $\xleftarrow{VS_j = V(E_j^t, pk; t_1, t_2, id, P_j)\{(h_j, \{d_k^j\}) \in R_s\}}$ | $E_j^t = Enc_{pk}(\{d_k^j\}_{k=1}^n)$

If VS is not received from at least one of the parties before time t_1

Resolve 1

$\xrightarrow{\{d_k^i\} j, PK(h_i, \{a_k\})\{(h_i, \{d_k^i\}) \in R_s\}}$

$\xleftarrow{\{d_k^j\}, PK(h_j, \{a_k\})\{(h_j, \{d_k^j\}) \in R_s\}}$

If d_k^i are not received before t_2

Resolve 2

Phase 3

Note that $\mathbf{a_k} = \mathbf{g^{r_k}}$ (First part of the k^{th} item's encryption).
The relation R_s is $\mathbf{\log_g h_i} = \mathbf{\log_{a_k} a_k^{x_i}}$ for each k.

$\underline{P_i}$	**Phase 3**	$\underline{P_j}$
compute $\{d_k^i = a_k^{x_i}\}_{k=1}^n$	$\xrightarrow{VS_i = V(E_i^t, pk; t_1, t_2, id, P_i)\{(h_i, \{d_k^i\}) \in R_s\}}$	compute $\{d_k^j = a_k^{x_j}\}_{k=1}^n$
$E_i^t = Enc_{pk}(\{d_k^i\}_{k=1}^n)$	$\xleftarrow{VS_j = V(E_j^t, pk; t_1, t_2, id, P_j)\{(h_j, \{d_k^j\}) \in R_s\}}$	$E_j^t = Enc_{pk}(\{d_k^j\}_{k=1}^n)$

If VS is not received from at least one of the parties before time t_1

Resolve 1

$$\xrightarrow{\{d_k^i\} j, PK(h_i, \{a_k\})\{(h_i, \{d_k^i\}) \in R_s\}}$$

$$\xleftarrow{\{d_k^j\}, PK(h_j, \{a_k\})\{(h_j, \{d_k^j\}) \in R_s\}}$$

If d_k^i are not received before t_2

Resolve 2

Phase 3

Note that $\mathbf{a_k} = \mathbf{g^{r_k}}$ (First part of the k^{th} item's encryption).
The relation R_s is $\mathbf{log_g h_i} = \mathbf{log_{a_k} a_k^{x_i}}$ for each k.

$\underline{P_i}$	**Phase 3**	$\underline{P_j}$
compute $\{d_k^i = a_k^{x_i}\}_{k=1}^n$	$\xrightarrow{VS_i = V(E_i^t, pk; t_1, t_2, id, P_i)\{(h_i, \{d_k^i\}) \in R_s\}}$	compute $\{d_k^j = a_k^{x_j}\}_{k=1}^n$
$E_i^t = Enc_{pk}(\{d_k^i\}_{k=1}^n)$	$\xleftarrow{VS_j = V(E_j^t, pk; t_1, t_2, id, P_j)\{(h_j, \{d_k^j\}) \in R_s\}}$	$E_j^t = Enc_{pk}(\{d_k^j\}_{k=1}^n)$

If VS is not received from at least one of the parties before time t_1

Resolve 1

$$\xrightarrow{\{d_k^i\}j, PK(h_i, \{a_k\})\{(h_i, \{d_k^i\}) \in R_s\}}$$

$$\xleftarrow{\{d_k^j\}, PK(h_j, \{a_k\})\{(h_j, \{d_k^j\}) \in R_s\}}$$

If d_k^i are not received before t_2

Resolve 2

Phase 3

Note that $\mathbf{a_k} = \mathbf{g^{r_k}}$ (First part of the k^{th} item's encryption).
The relation R_s is $\mathbf{\log_g h_i} = \mathbf{\log_{a_k} a_k^{x_i}}$ for each k.

Phase 3

$\underline{P_i}$		$\underline{P_j}$
compute $\{d_k^i = a_k^{x_i}\}_{k=1}^n$	$\xrightarrow{VS_i = V(E_i^t, pk; t_1, t_2, id, P_i)\{(h_i, \{d_k^i\}) \in R_s\}}$	compute $\{d_k^j = a_k^{x_j}\}_{k=1}^n$
$E_i^t = Enc_{pk}(\{d_k^i\}_{k=1}^n)$	$\xleftarrow{VS_j = V(E_j^t, pk; t_1, t_2, id, P_j)\{(h_j, \{d_k^j\}) \in R_s\}}$	$E_j^t = Enc_{pk}(\{d_k^j\}_{k=1}^n)$

If VS is not received from at least one of the parties before time t_1

Resolve 1

$$\xrightarrow{\{d_k^i\}j, PK(h_i, \{a_k\})\{(h_i, \{d_k^i\}) \in R_s\}}$$

$$\xleftarrow{\{d_k^j\}, PK(h_j, \{a_k\})\{(h_j, \{d_k^j\}) \in R_s\}}$$

If d_k^i are not received before t_2

Resolve 2

Phase 3

Note that $\mathbf{a_k} = \mathbf{g^{r_k}}$ (First part of the k^{th} item's encryption).
The relation R_s is $\mathbf{\log_g h_i} = \mathbf{\log_{a_k} a_k^{x_i}}$ for each k.

$\underline{P_i}$	**Phase 3**	$\underline{P_j}$
compute $\{d_k^i = a_k^{x_i}\}_{k=1}^n$	$\xrightarrow{VS_i = V(E_i^t, pk; t_1, t_2, id, P_i)\{(h_i, \{d_k^i\}) \in R_s\}}$	compute $\{d_k^j = a_k^{x_j}\}_{k=1}^n$
$E_i^t = Enc_{pk}(\{d_k^i\}_{k=1}^n)$	$\xleftarrow{VS_j = V(E_j^t, pk; t_1, t_2, id, P_j)\{(h_j, \{d_k^j\}) \in R_s\}}$	$E_j^t = Enc_{pk}(\{d_k^j\}_{k=1}^n)$

If VS is not received from at least one of the parties before time t_1

Resolve 1

$$\xrightarrow{\{d_k^i\}j, PK(h_i, \{a_k\})\{(h_i, \{d_k^i\}) \in R_s\}}$$

$$\xleftarrow{\{d_k^j\}, PK(h_j, \{a_k\})\{(h_j, \{d_k^j\}) \in R_s\}}$$

If d_k^i are not received before t_2

Resolve 2