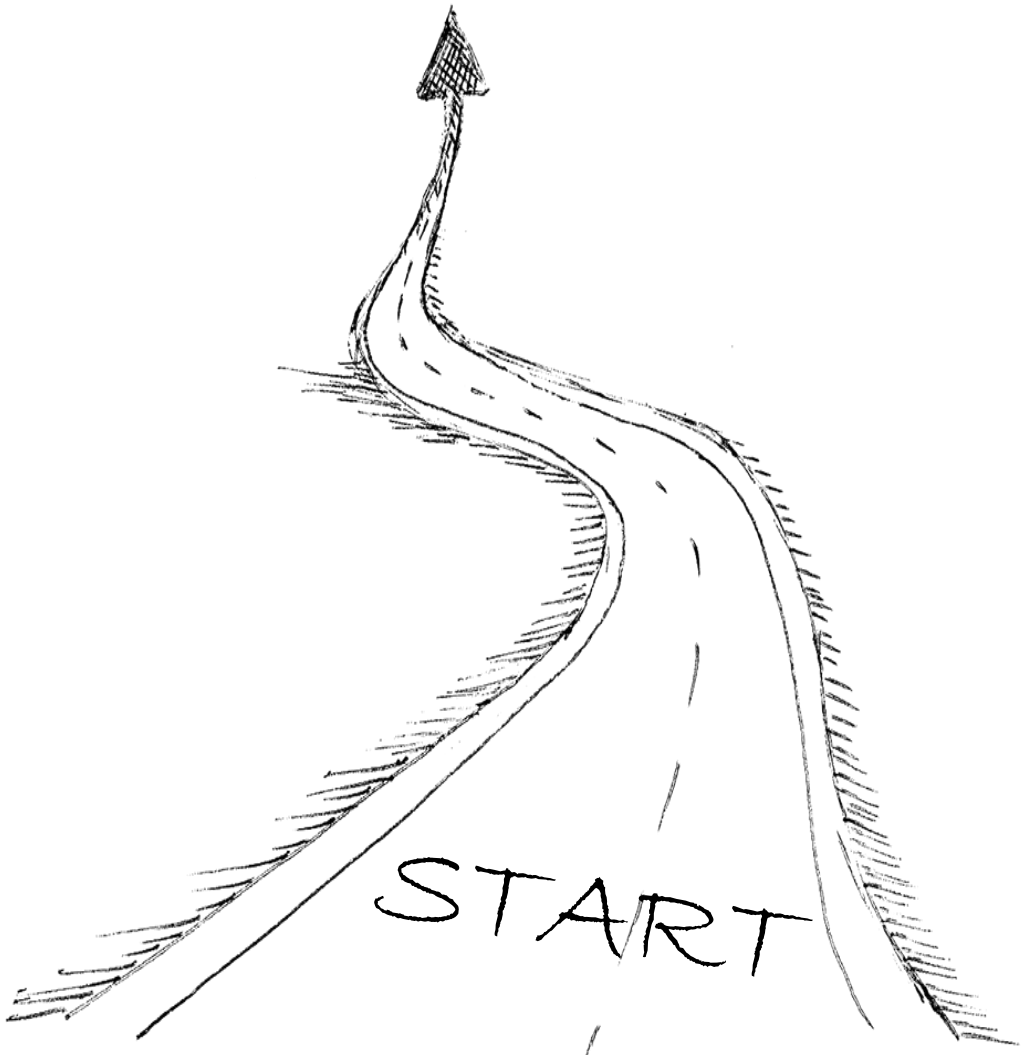
There Is Always an Exception: Controlling Partial Information Leakage in Secure Computation

ICISC-2019

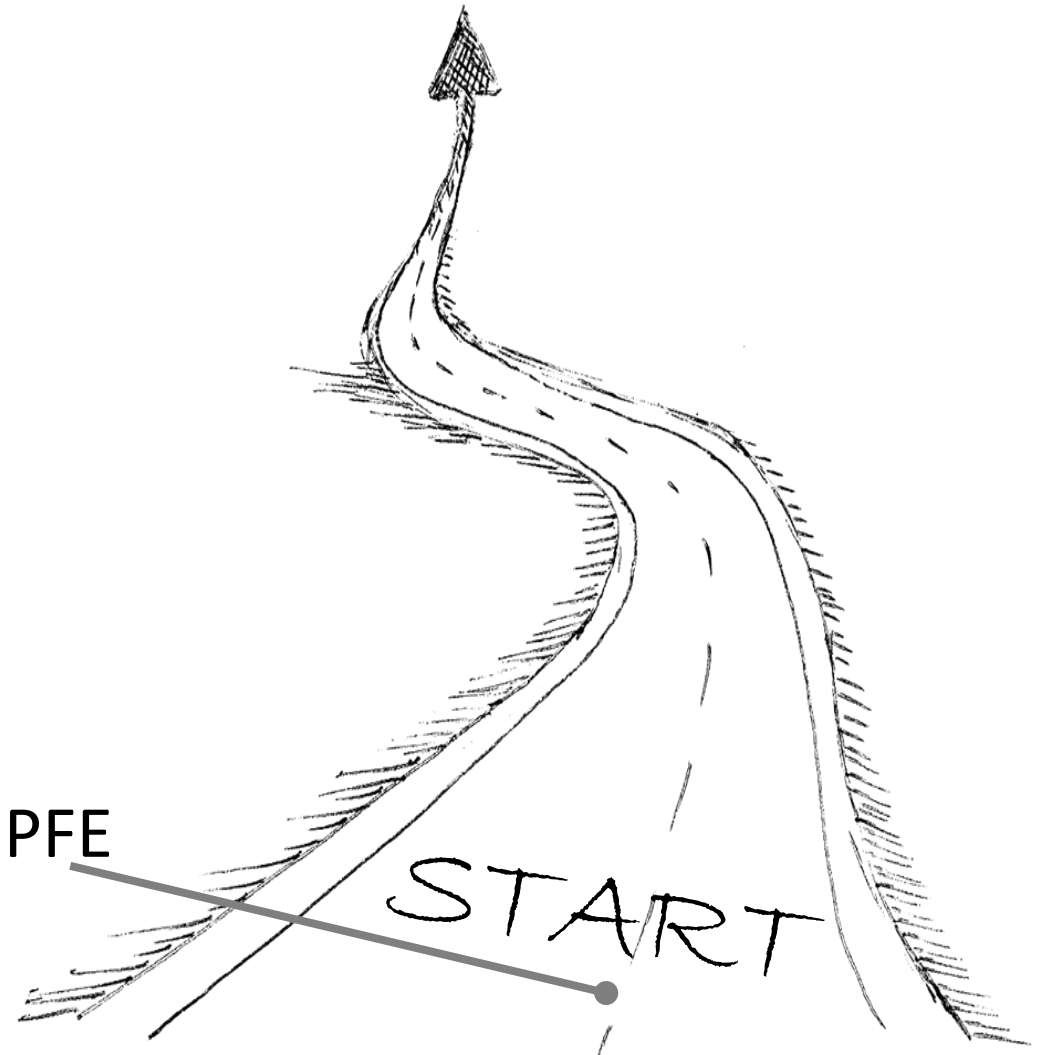
M. Horváth, L. Buttyán, G. Székely, D. Neubrandt

Budapest University of Technology and Economics,
Laboratory of Cryptography and System Security

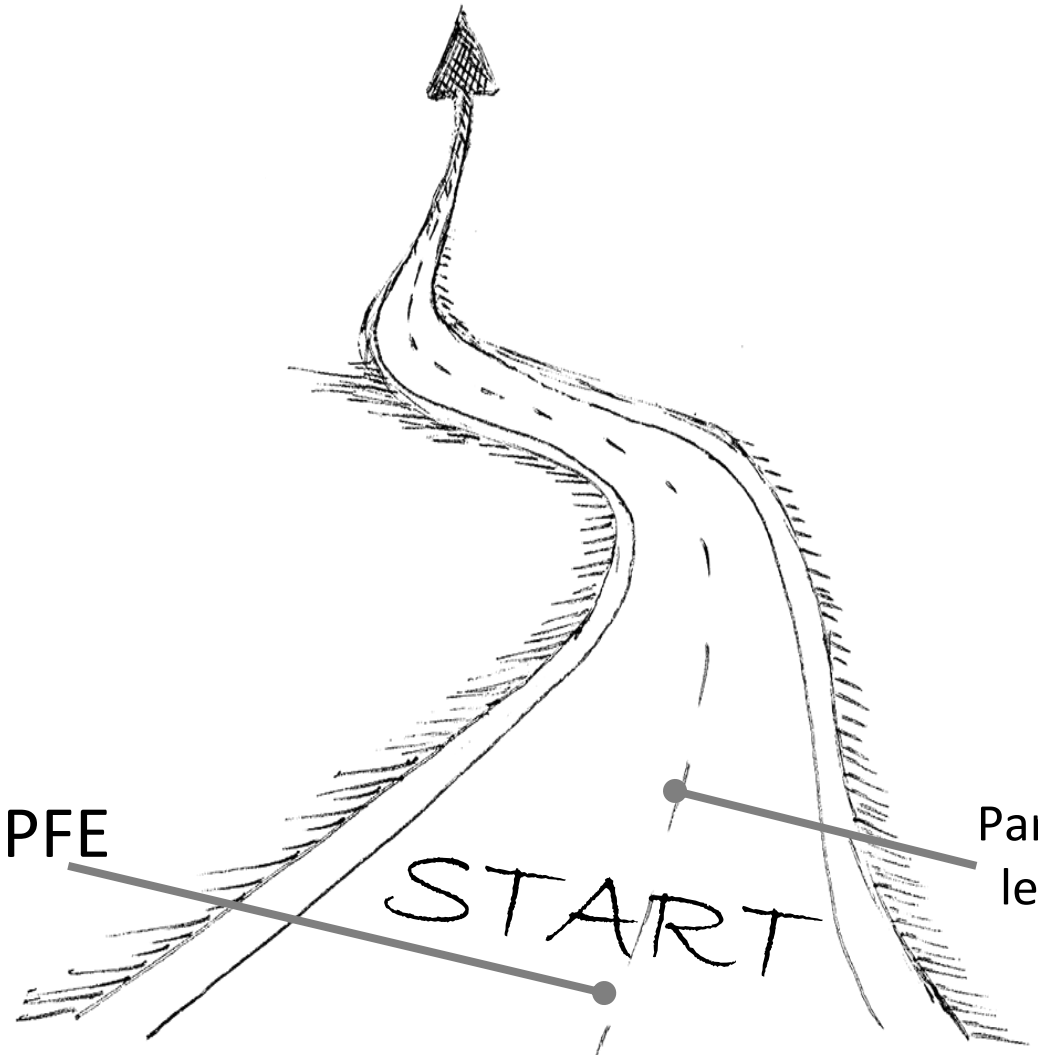
Roadmap



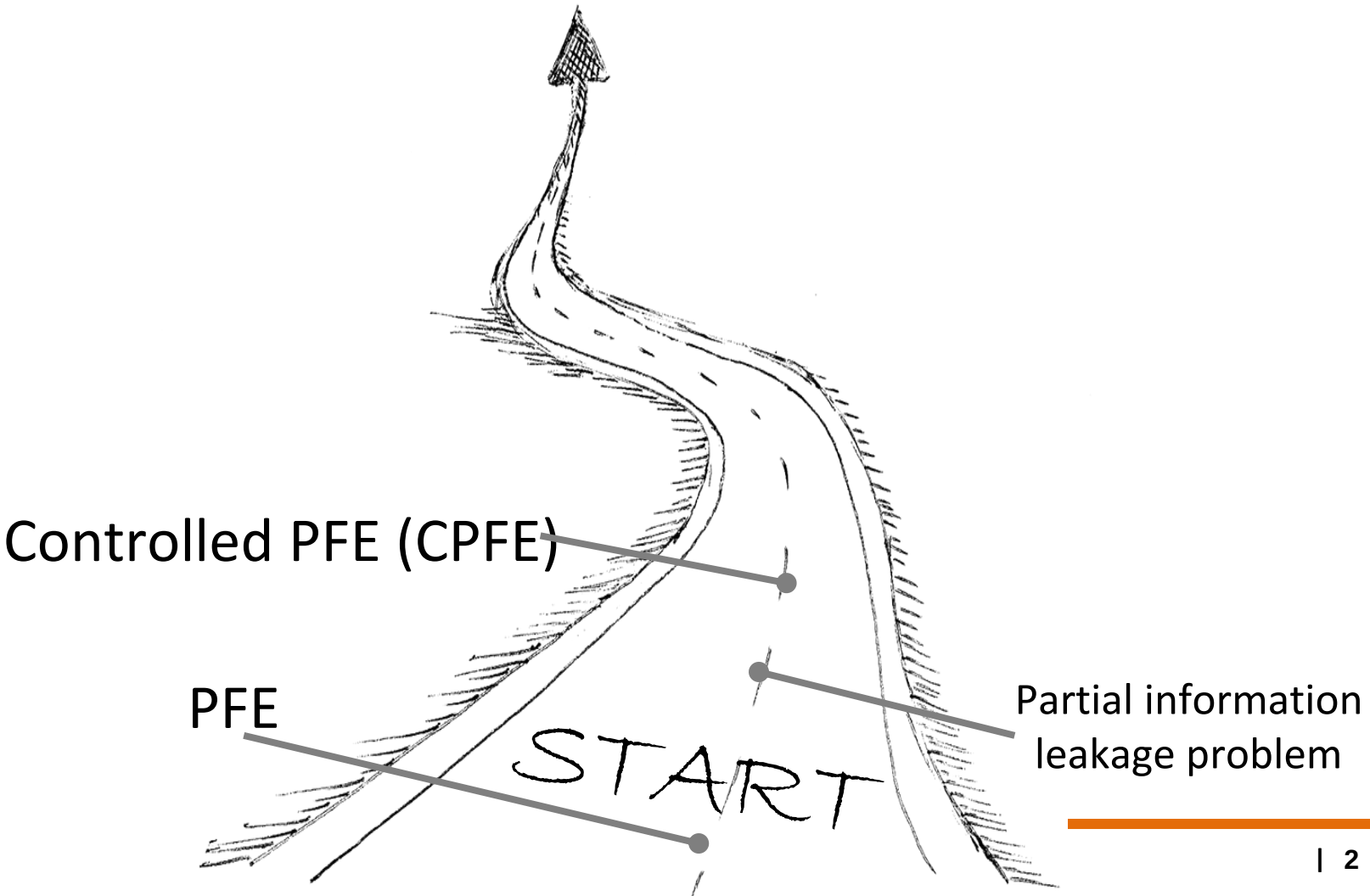
Roadmap



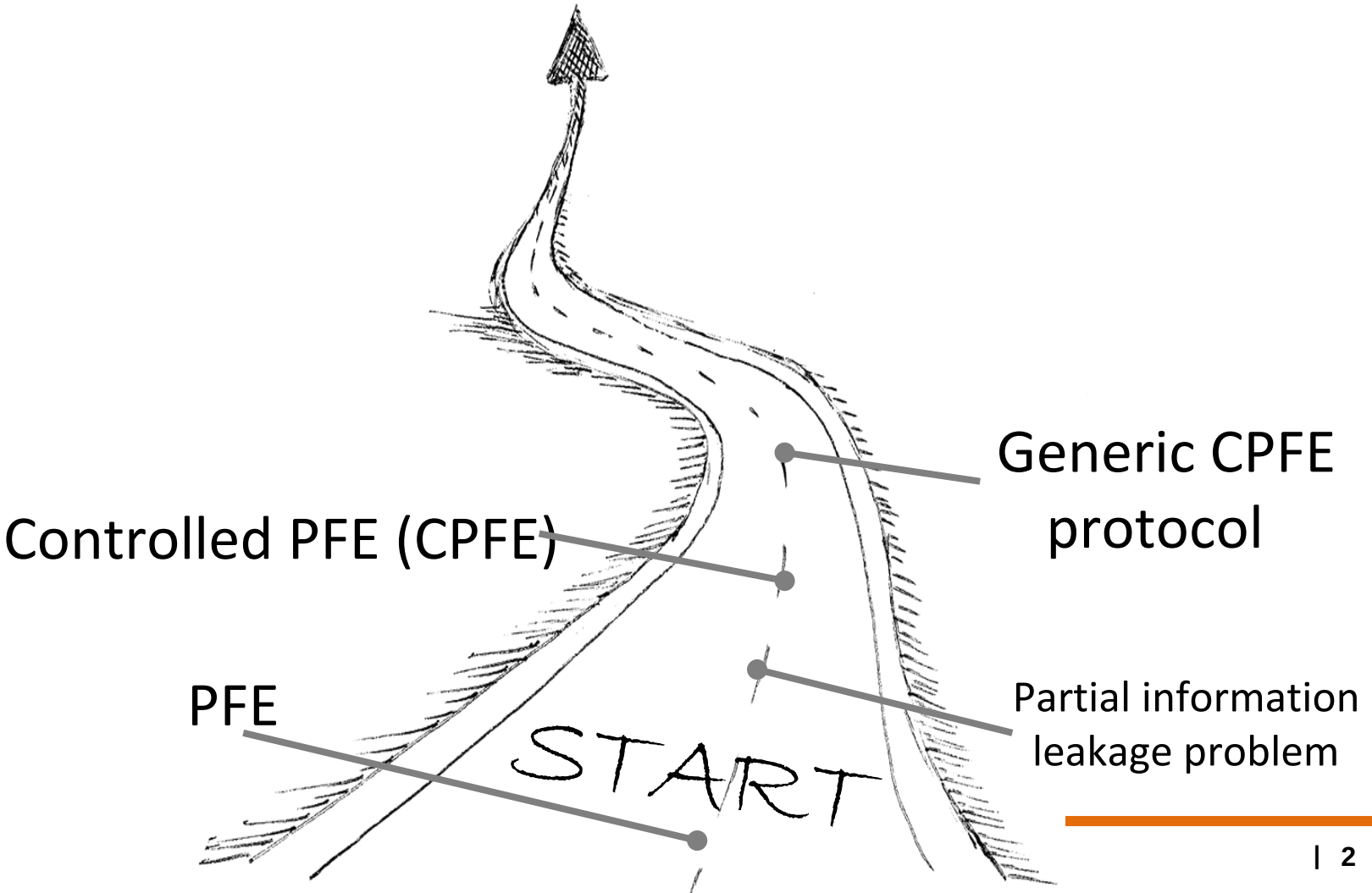
Roadmap



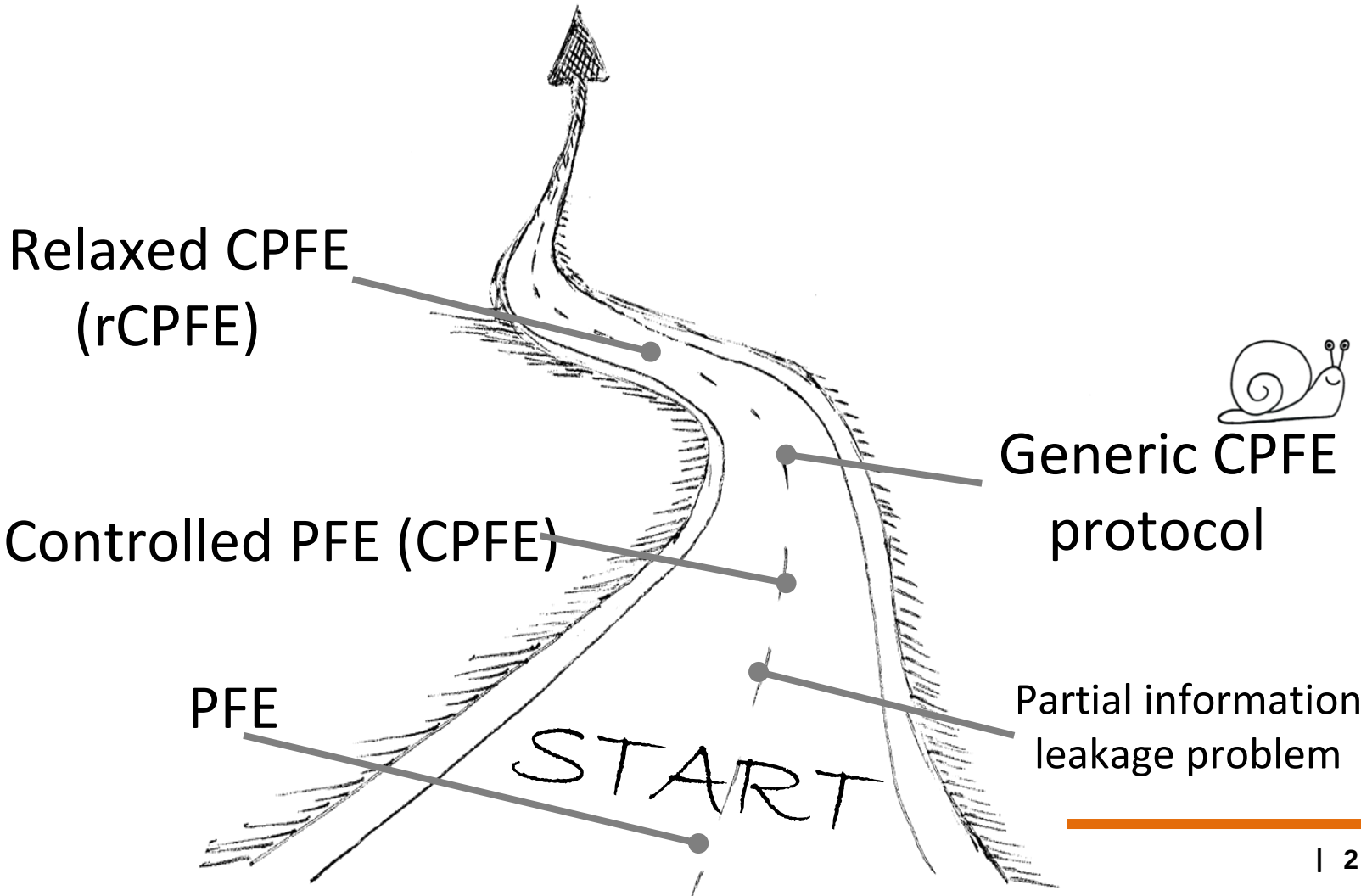
Roadmap


Controlled PFE (CPFE)
PFE
START
Partial information leakage problem

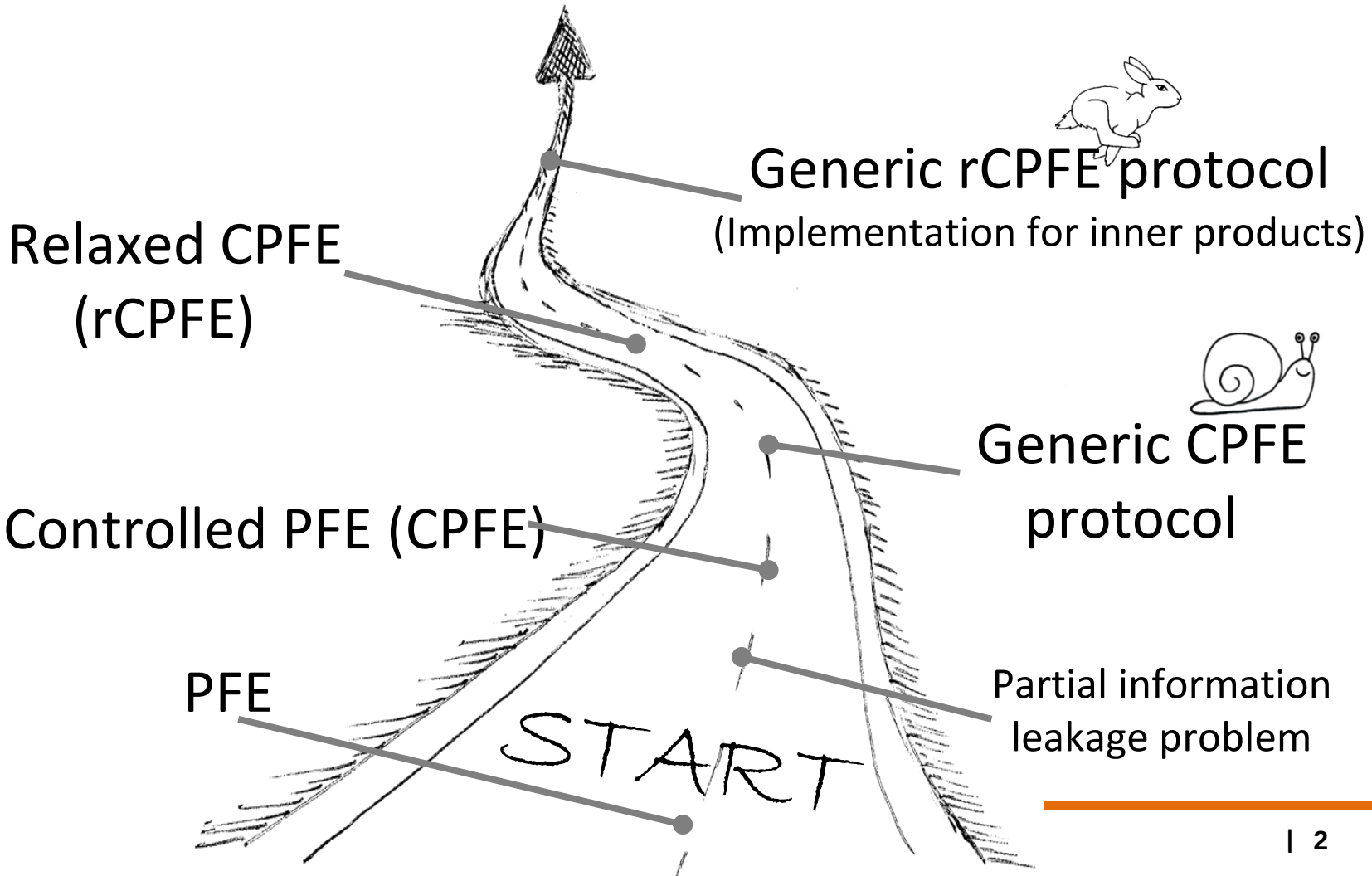
Roadmap


Generic CPFE protocol
Controlled PFE (CPFE)
PFE
START
Partial information leakage problem

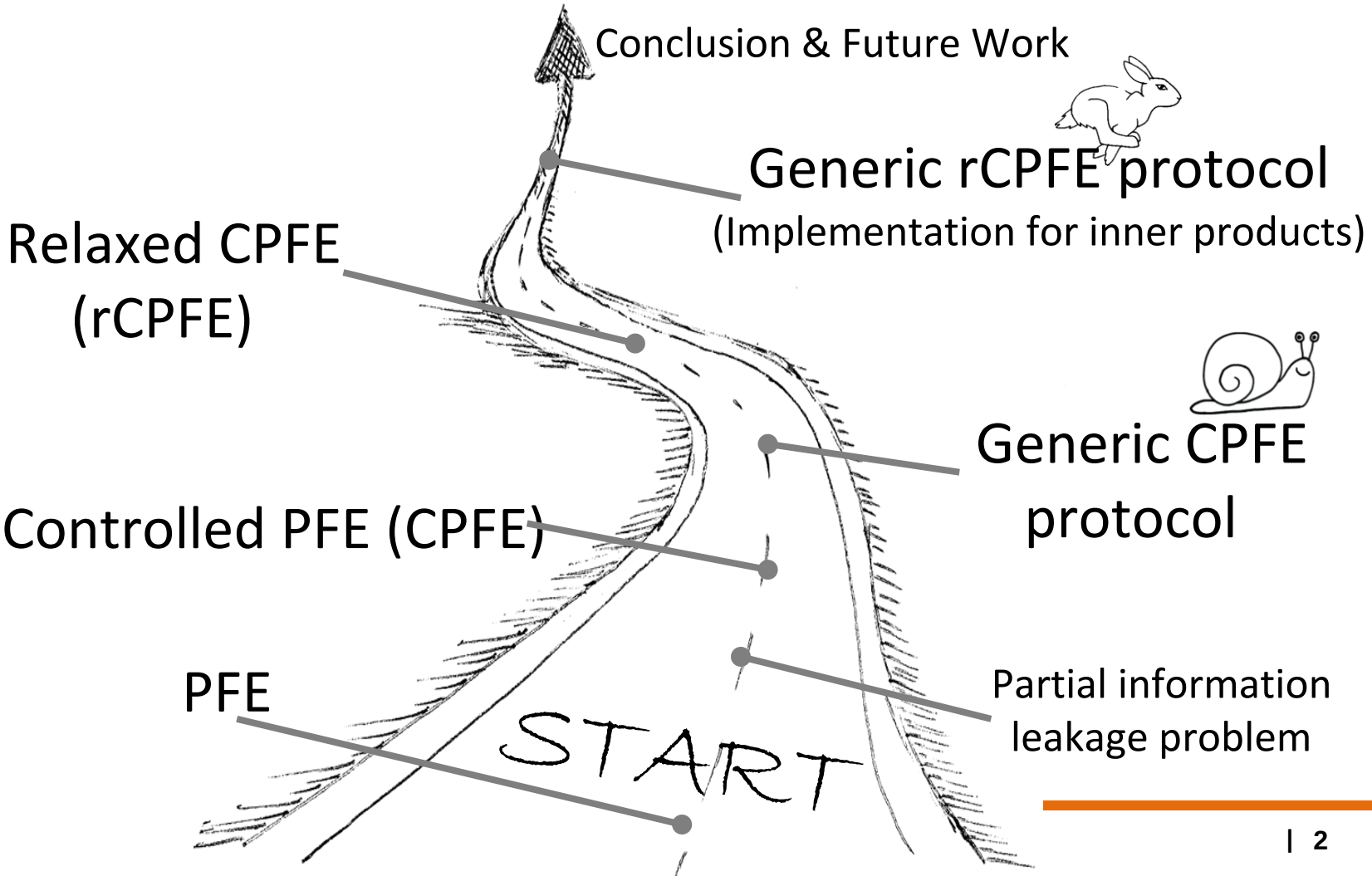
Roadmap


Relaxed CPFE
(rCPFE)
Generic CPFE
protocol
Controlled PFE (CPFE)
PFE
START
Partial information
leakage problem

Roadmap


Generic rCPFE protocol
(Implementation for inner products)
Relaxed CPFE
(rCPFE)
Generic CPFE
protocol
Controlled PFE (CPFE)
PFE
Partial information
leakage problem
START

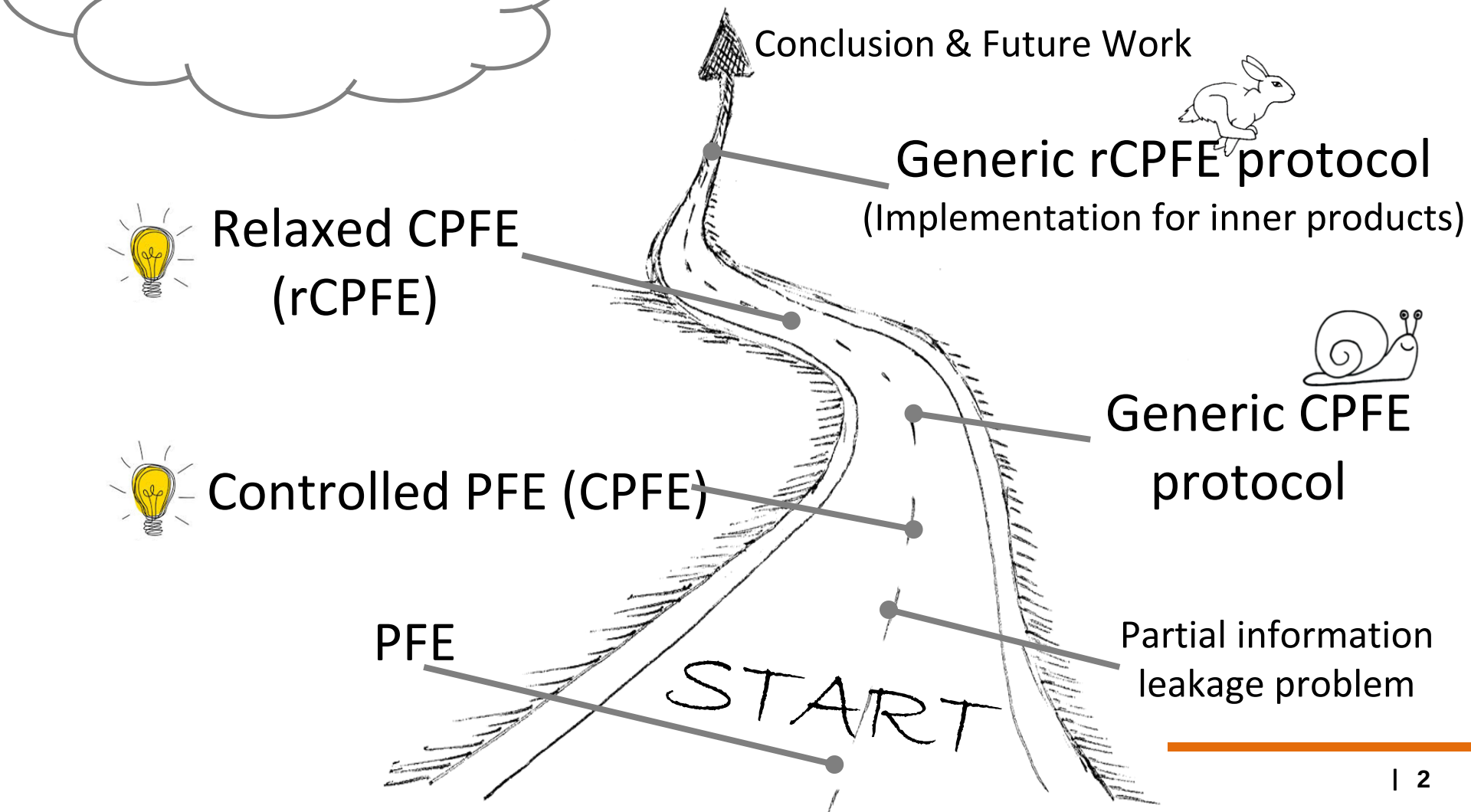
Roadmap


Conclusion & Future Work
Generic rCPFE protocol
(Implementation for inner products)
Relaxed CPFE
(rCPFE)
Generic CPFE
protocol
Controlled PFE (CPFE)
PFE
Partial information
leakage problem
START

Roadmap

Contributions:

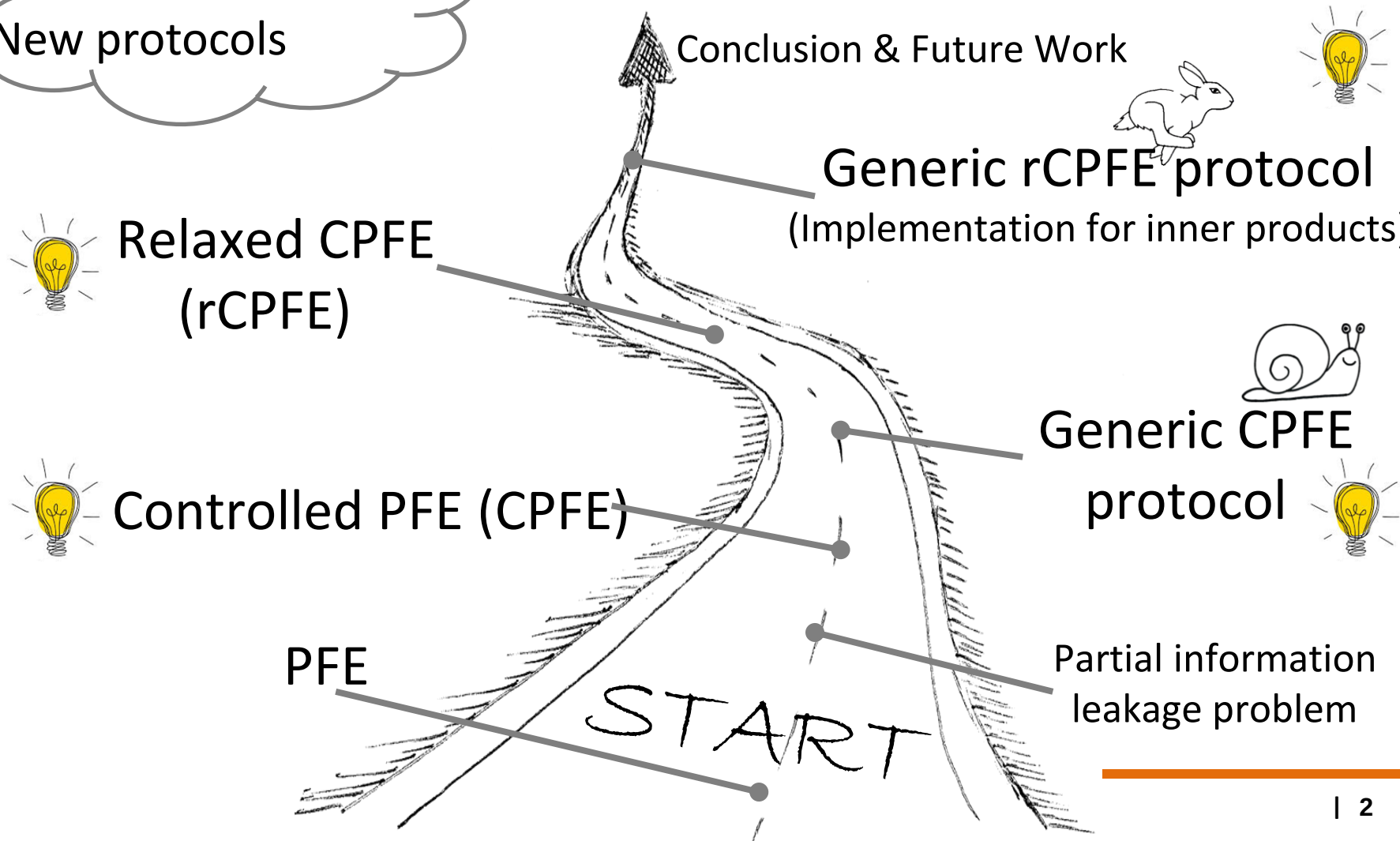
- Definitional framework



Roadmap

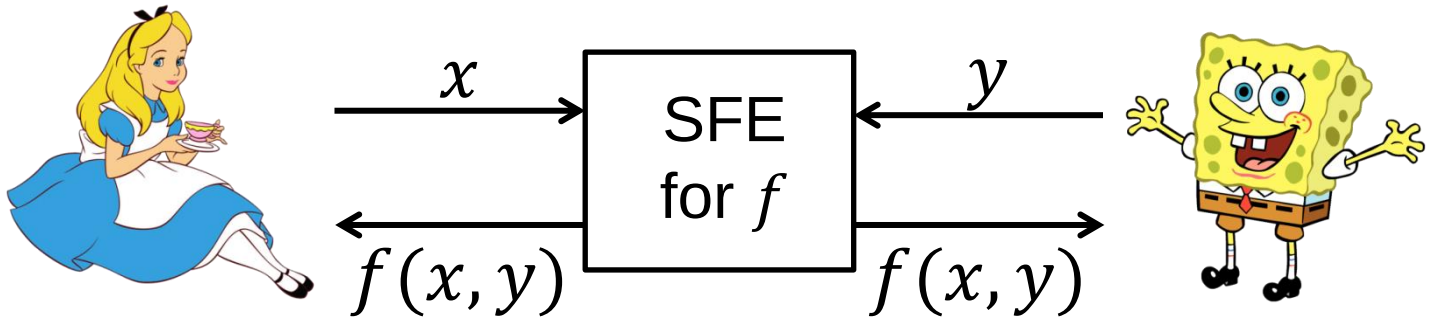
Contributions:

- Definitional framework
- New protocols



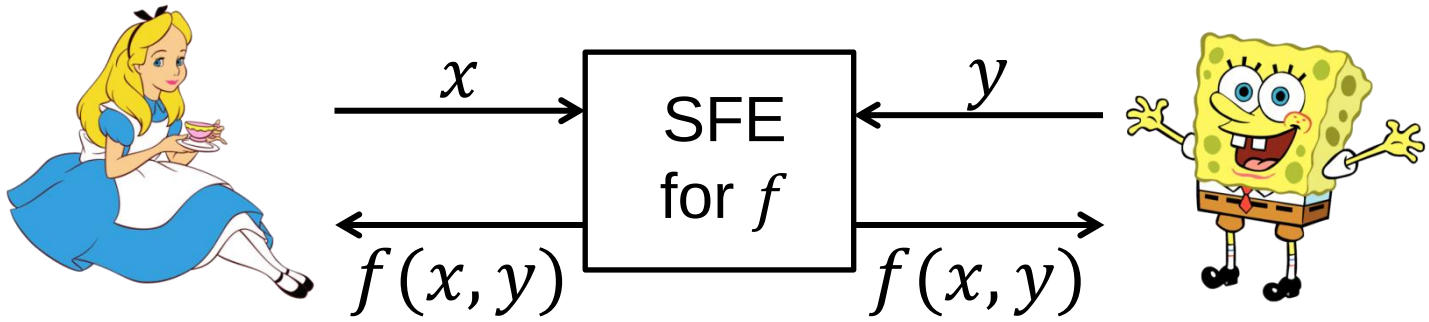
Background

- *Secure* Function Evaluation (SFE or 2PC)

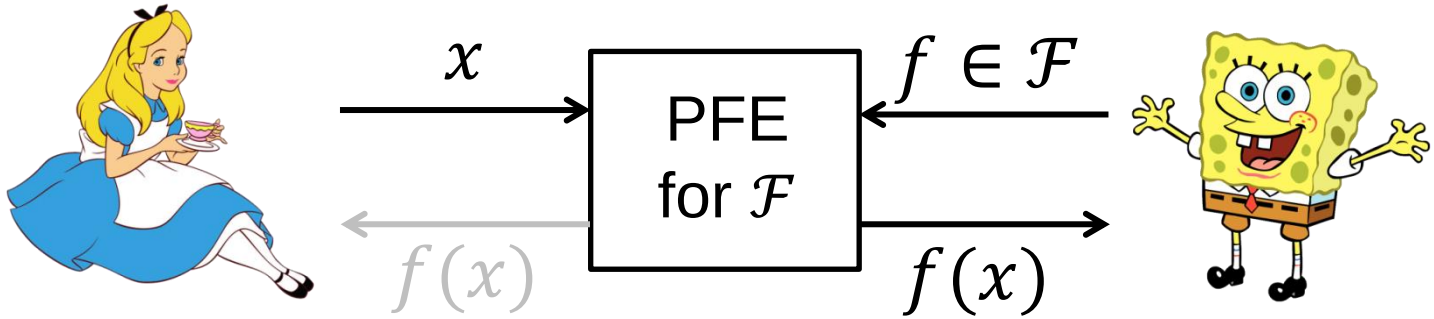


Background

- *Secure* Function Evaluation (SFE or 2PC)

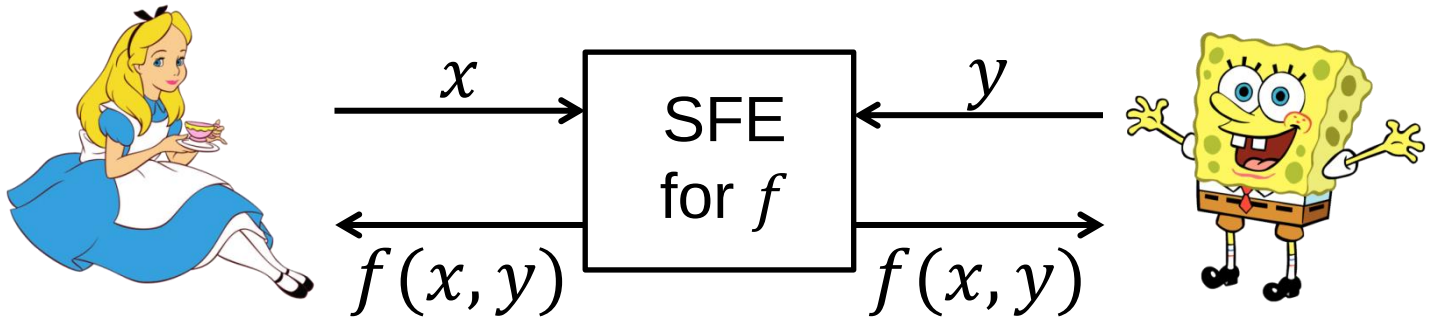


- *Private* Function Evaluation (PFE)

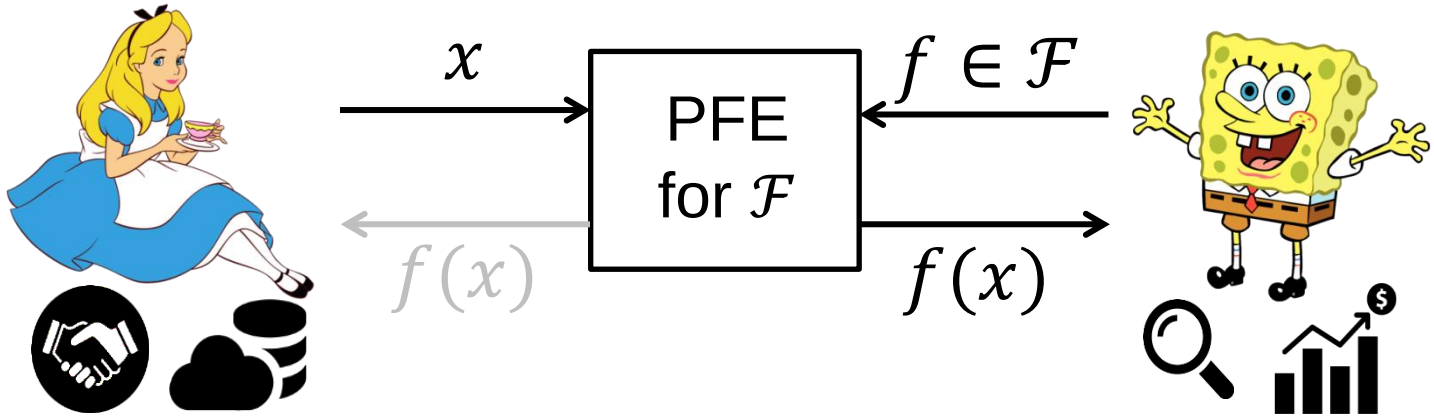


Background

- *Secure* Function Evaluation (SFE or 2PC)



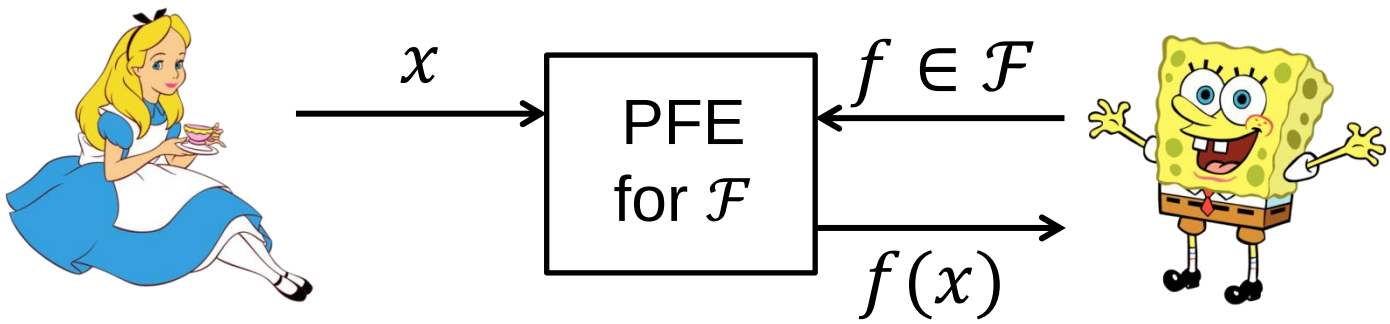
- *Private* Function Evaluation (PFE)



Data Broker

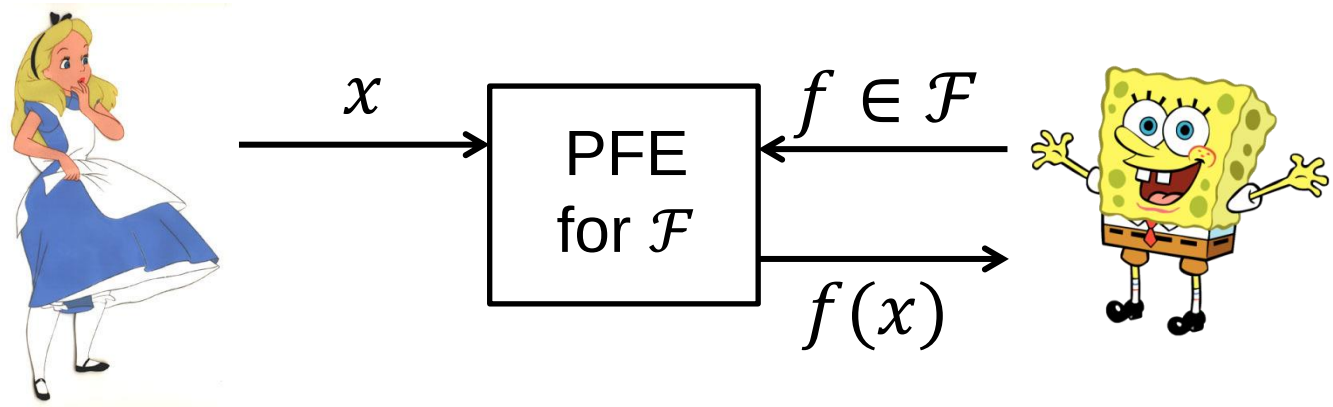
Value-Added Service Provider

Problem Statement



- Requirements from PFE:
  - **Function privacy**: *Alice does no learn anything about* f
  - **Data privacy**: *Bob does not learn more about* x *than what is already revealed by* $f(x)$

Problem Statement



- Requirements from PFE:
  - **Function privacy**: *Alice does no learn anything about f*
  - **Data privacy**: *Bob does not learn more about x than what is already revealed by $f(x)$*

- What if besides x, some partial information $g(x)$ is also sensitive?
- In particular, what if $g \in \mathcal{F}$ as well?

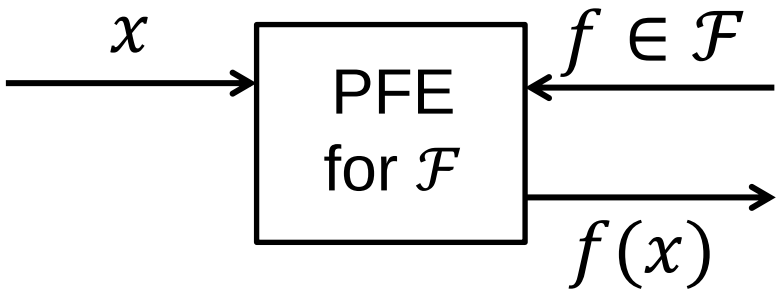
Example: Privacy-Preserving Credit Checking

- Is Alice eligible for credit in the Bank of Korea? [PSS09]

한국은행
THE BANK OF KOREA

Example: Privacy-Preserving Credit Checking

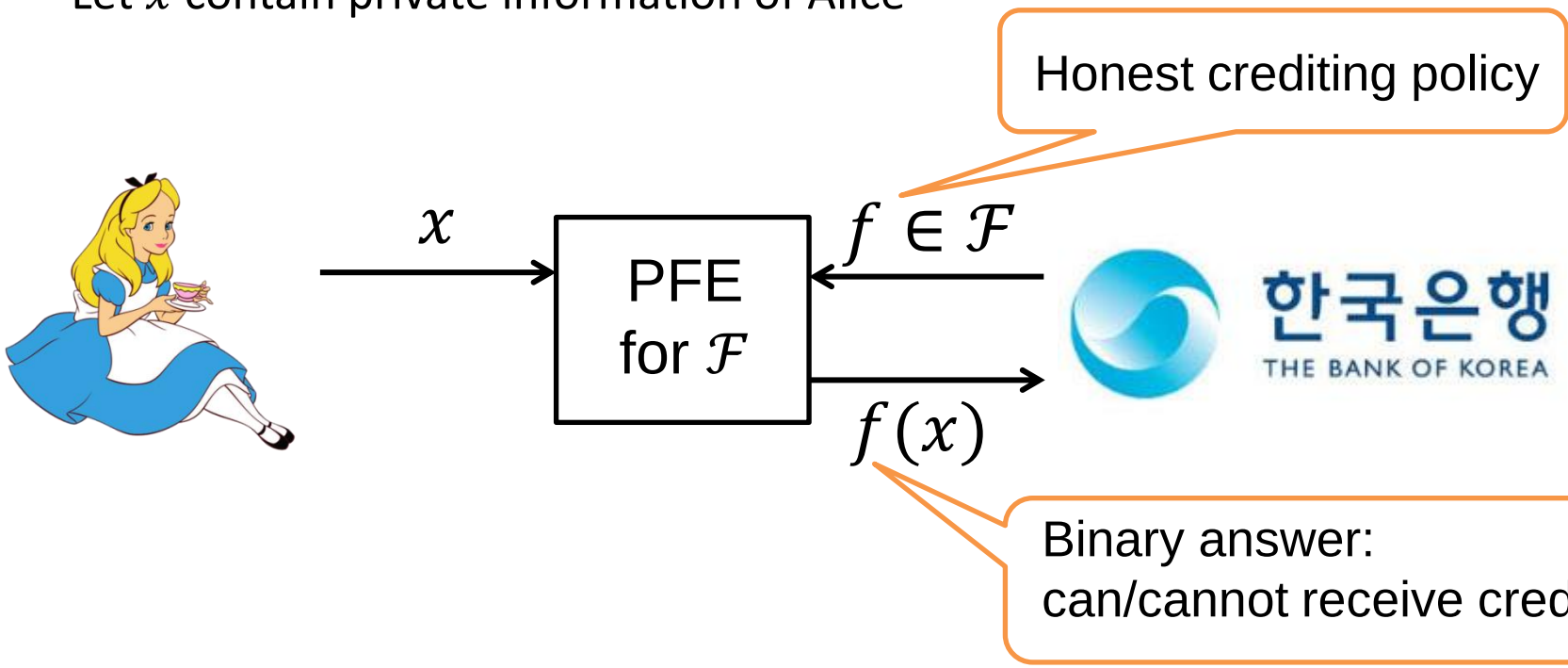
- Is Alice eligible for credit in the Bank of Korea? [PSS09]
  - Let $\mathcal{F}$ be the class of Boolean functions.
  - Let $f \in \mathcal{F}$ represent the crediting policy of the bank.
  - Let x contain private information of Alice



한국은행
THE BANK OF KOREA

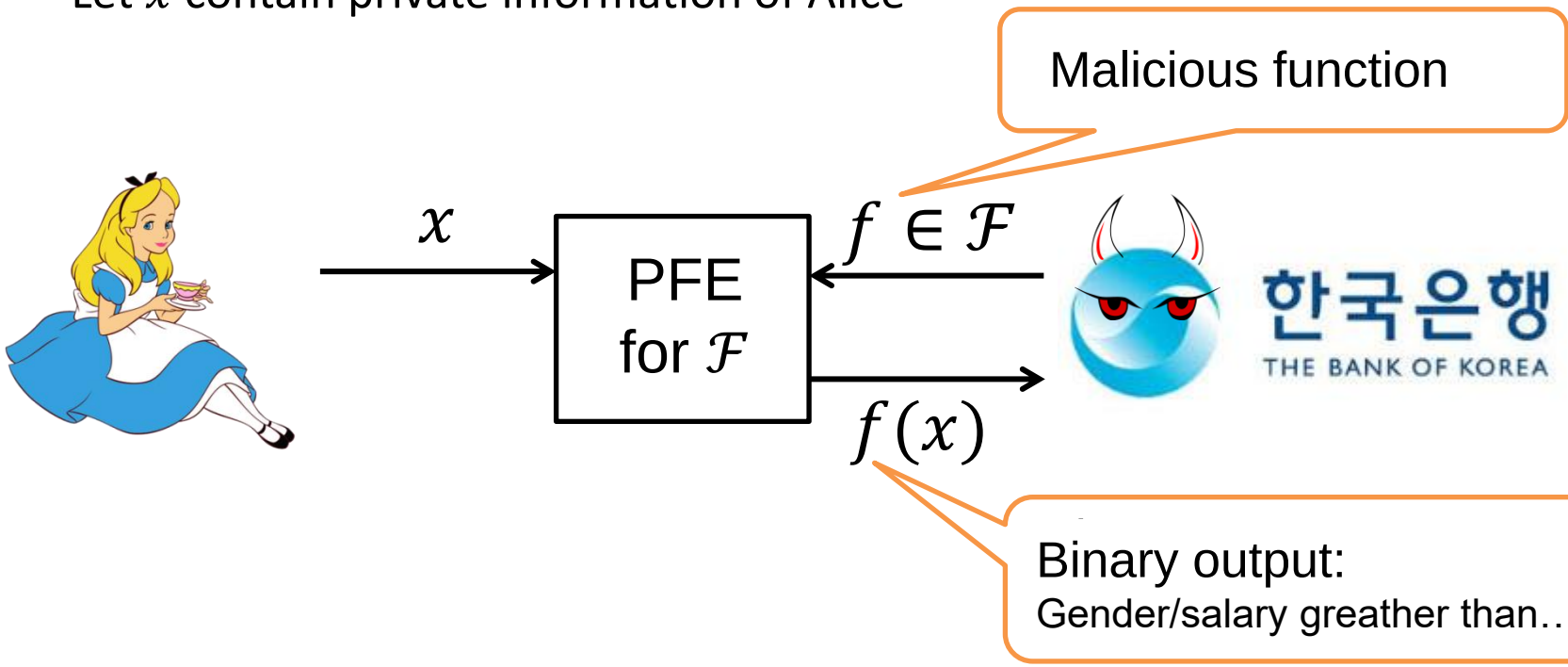
Example: Privacy-Preserving Credit Checking

- Is Alice eligible for credit in the Bank of Korea? [PSS09]
  - Let $\mathcal{F}$ be the class of Boolean functions.
  - Let $f \in \mathcal{F}$ represent the crediting policy of the bank.
  - Let x contain private information of Alice



Example: Privacy-Preserving Credit Checking

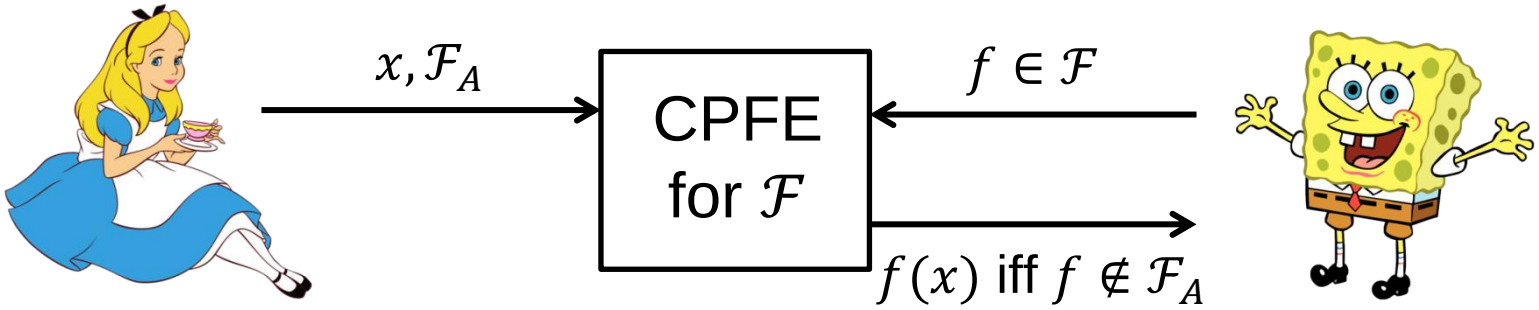
- Is Alice eligible for credit in the Bank of Korea? [PSS09]
  - Let $\mathcal{F}$ be the class of Boolean functions.
  - Let $f \in \mathcal{F}$ represent the crediting policy of the bank.
  - Let x contain private information of Alice



Controlled Private Function Evaluation (CPFE)

- Goal: To enable Alice to make exceptions, i.e. to define a set of forbidden functions $\mathcal{F}_A \subset \mathcal{F}$

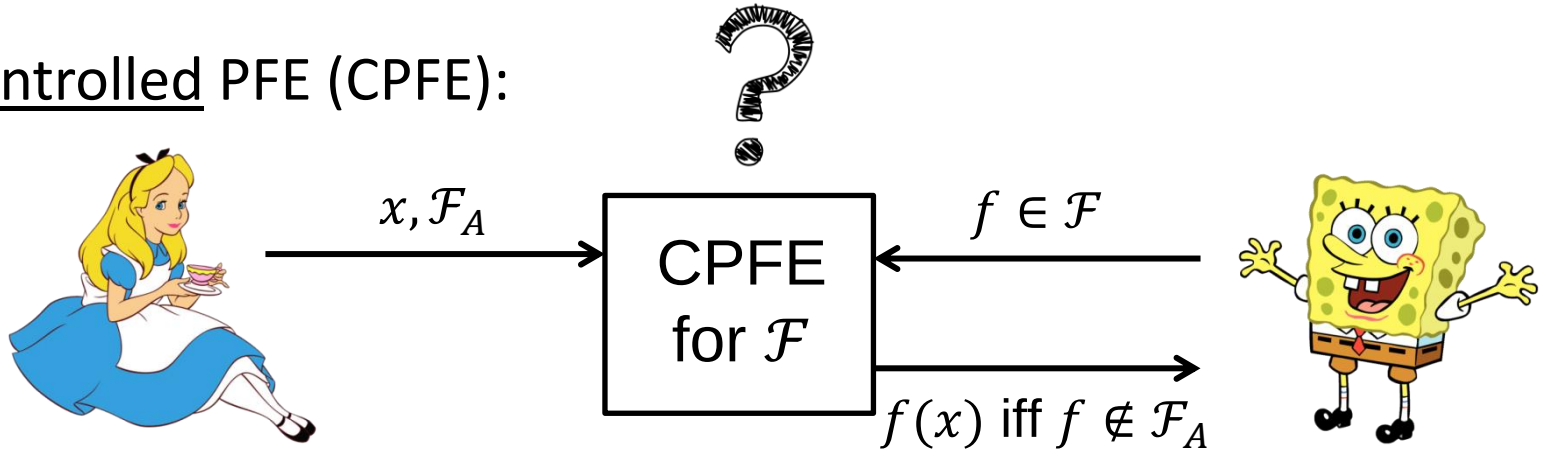
- Controlled PFE (CPFE):



Controlled Private Function Evaluation (CPFE)

- Goal: To enable Alice to make exceptions, i.e. to define a set of forbidden functions $\mathcal{F}_A \subset \mathcal{F}$

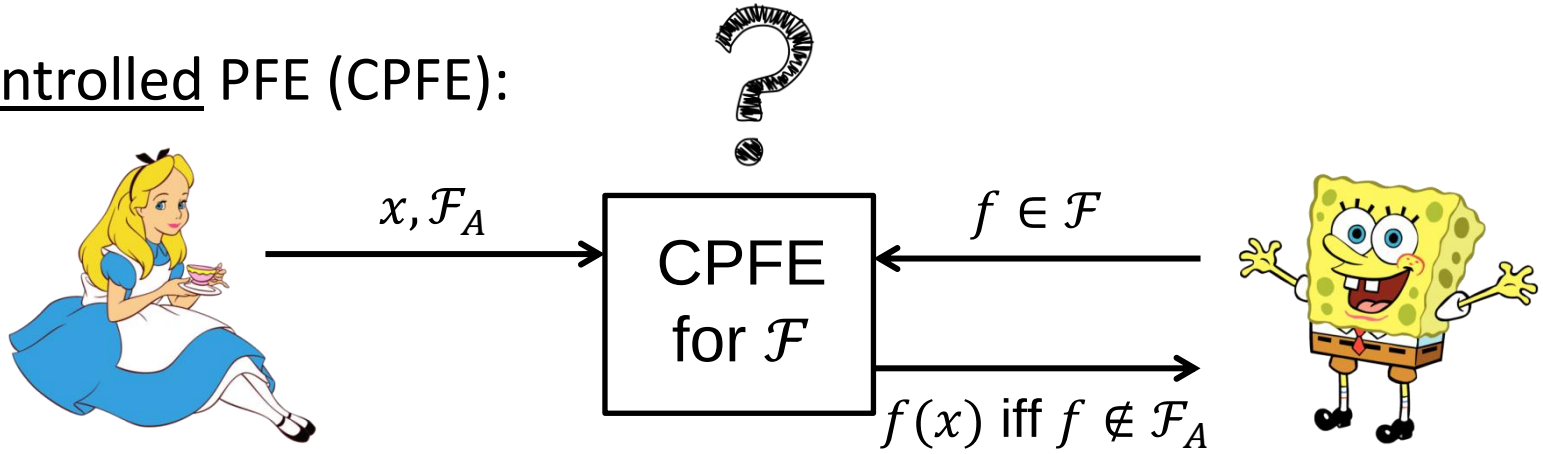
- Controlled PFE (CPFE):



Controlled Private Function Evaluation (CPFE)

- Goal: To enable Alice to make exceptions, i.e. to define a set of forbidden functions $\mathcal{F}_A \subset \mathcal{F}$

- Controlled PFE (CPFE):



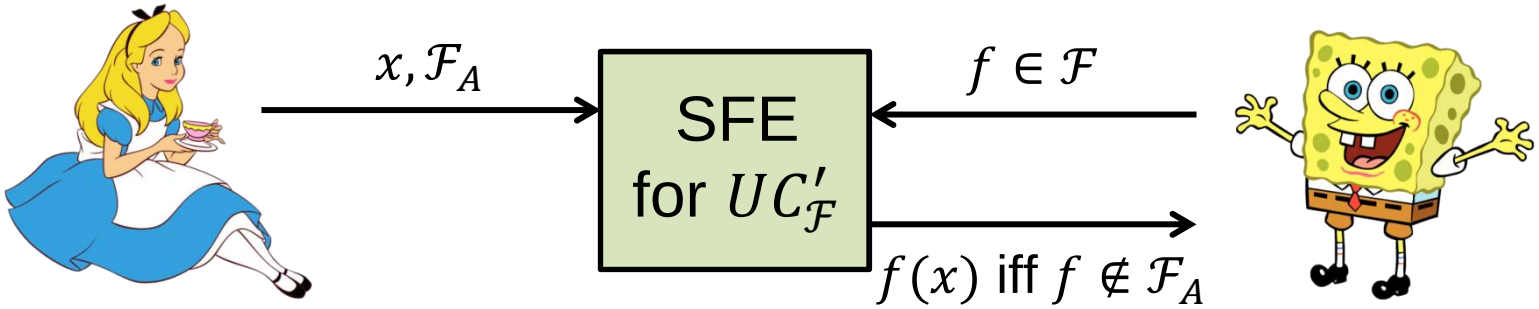
- Realization idea: with SFE and conditional universal circuits
- Universal circuit (UC) is a ``programable'' function for $\mathcal{F}$:

$$UC_{\mathcal{F}}(f, x) = f(x)$$

Controlled Private Function Evaluation (CPFE)

- Goal: To enable Alice to make exceptions, i.e. to define a set of forbidden functions $\mathcal{F}_A \subset \mathcal{F}$

- Controlled PFE (CPFE):



- Realization idea: with SFE and conditional universal circuits
- Universal circuit (UC) is a ``programable'' function for $\mathcal{F}$:

$$UC_{\mathcal{F}}(f, x) = f(x)$$

- Conditional UC:

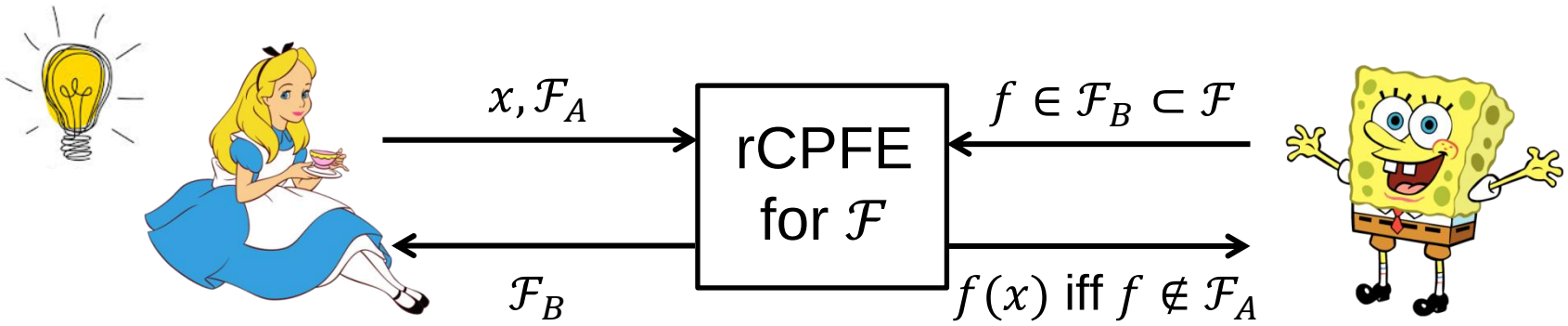
$$UC'_{\mathcal{F}}(f, x, \mathcal{F}_A) = \begin{cases} f(x) \text{ iff } f \notin \mathcal{F}_A \\ \perp \text{ otherwise} \end{cases}$$

Towards an Efficiency-Function Privacy Trade-off

- Drawbacks of the previous solution:
  - UCs are not efficient enough (even for a single evaluation)
  - Scalability: evaluating the same f on d different inputs costs d times more for both parties
- Goal: *to relax the function privacy requirement to achieve additive online overhead in case of multiple evaluations of* f

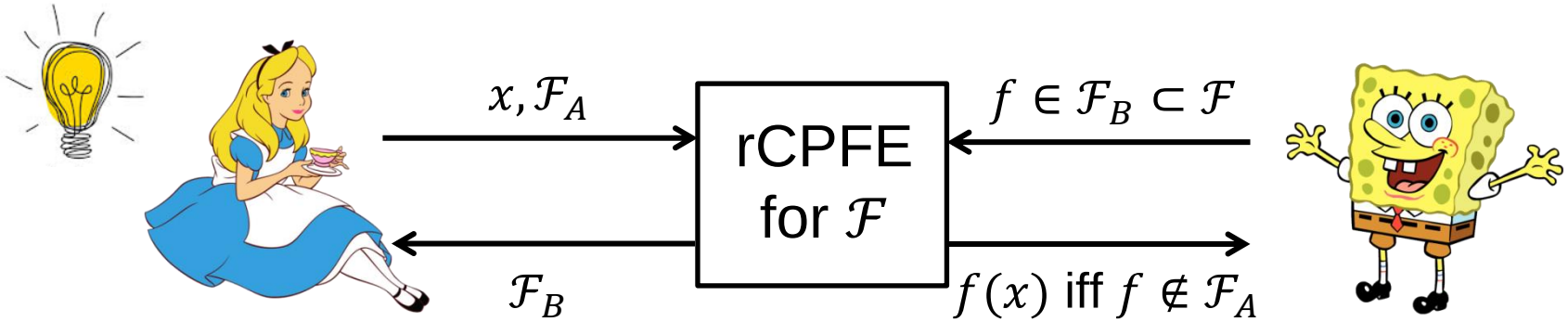
Towards an Efficiency-Function Privacy Trade-off

- Drawbacks of the previous solution:
  - UCs are not efficient enough (even for a single evaluation)
  - Scalability: evaluating the same f on d different inputs costs d times more for both parties
- Goal: *to relax the function privacy requirement to achieve additive online overhead in case of multiple evaluations of* f
- *k-relaxed CPFE (rCPFE) for* $|\mathcal{F}_B| = k$:



Towards an Efficiency-Function Privacy Trade-off

- Drawbacks of the previous solution:
  - UCs are not efficient enough (even for a single evaluation)
  - Scalability: evaluating the same f on d different inputs costs d times more for both parties
- Goal: *to relax the function privacy requirement to achieve additive online overhead in case of multiple evaluations of* f
- *k-relaxed CPFE (rCPFE) for* $|\mathcal{F}_B| = k$:

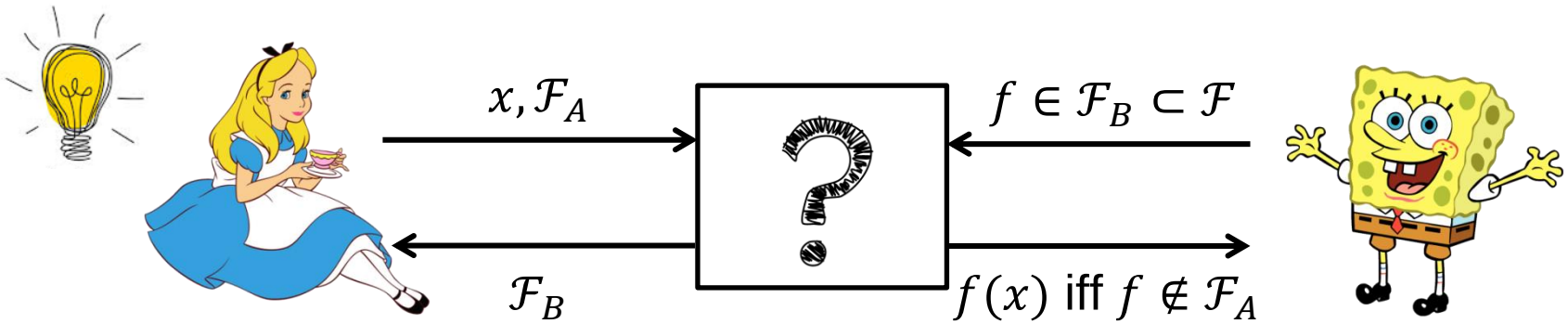


- Strong *k-rCPFE: extra requirement*

$$\forall\ PPT\ \mathcal{A}\colon\ P(\mathcal{A}(aux, \mathcal{F}_B) = f) - \frac{1}{k} < negl(\lambda)$$

Towards an Efficiency-Function Privacy Trade-off

- Drawbacks of the previous solution:
  - UCs are not efficient enough (even for a single evaluation)
  - Scalability: evaluating the same f on d different inputs costs d times more for both parties
- Goal: *to relax the function privacy requirement to achieve additive online overhead in case of multiple evaluations of* f
- *k-relaxed CPFE (rCPFE) for* $|\mathcal{F}_B| = k$:

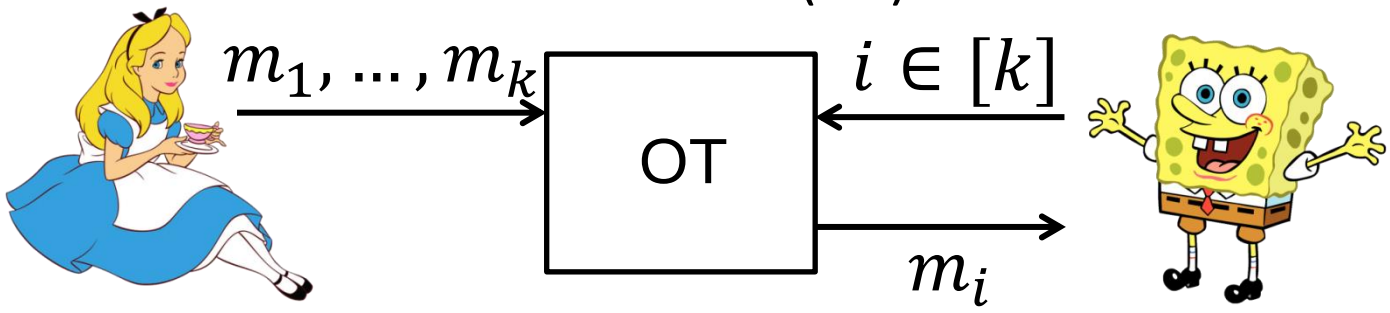


- Strong k-rCPFE: *extra requirement*

$$\forall\ PPT\ \mathcal{A}\colon\ P(\mathcal{A}(aux, \mathcal{F}_B) = f) - \frac{1}{k} < negl(\lambda)$$

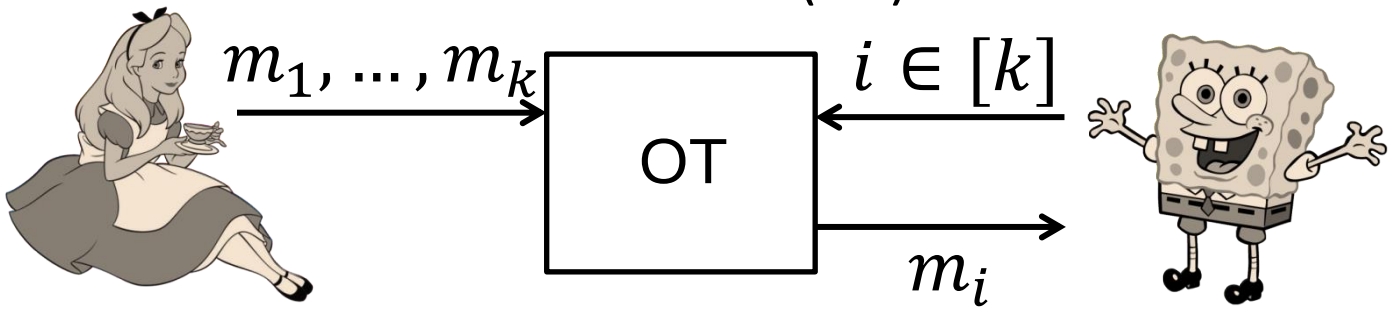
Naive OT-based Idea to Realise rCPFE

- A tool: 1 out of k Oblivious Transfer (OT):



Naive OT-based Idea to Realise rCPFE

- A tool: 1 out of k Oblivious Transfer (OT):

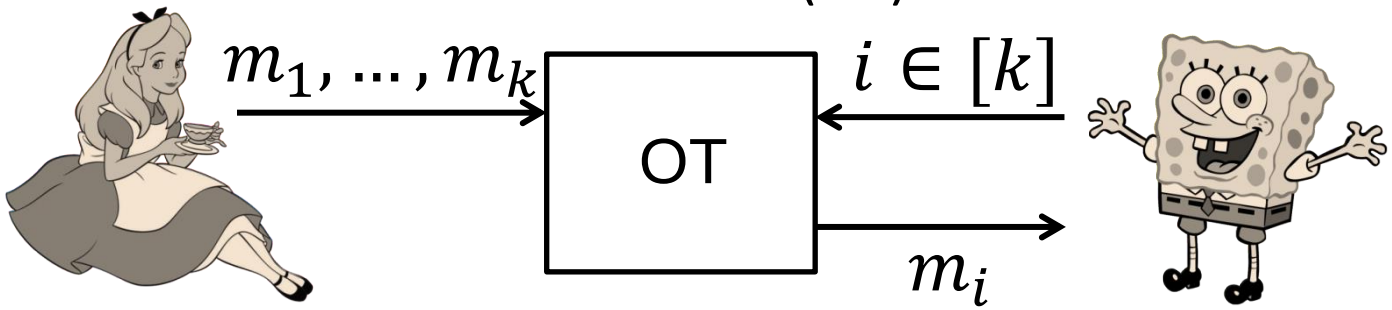


- A naive rCPFE protocol:

$x, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

Naive OT-based Idea to Realise rCPFE

- A tool: 1 out of k Oblivious Transfer (OT):



- A naive rCPFE protocol:

$x, \mathcal{F}_A$ — *INPUTS* — $f \in \mathcal{F}$

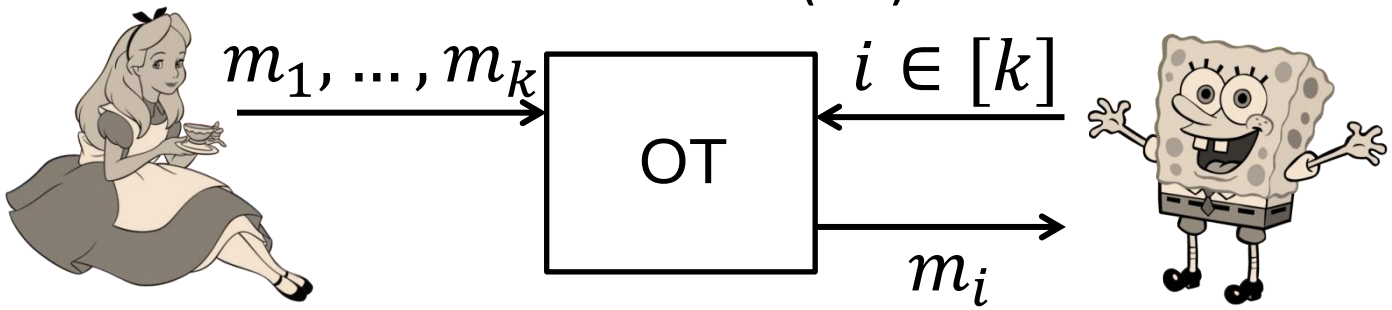
Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B$ ←

$\mathcal{F}_B := (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

Naive OT-based Idea to Realise rCPFE

- A tool: 1 out of k Oblivious Transfer (OT):



- A naive rCPFE protocol:

$x, \mathcal{F}_A$ — *INPUTS* — $f \in \mathcal{F}$

Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B$

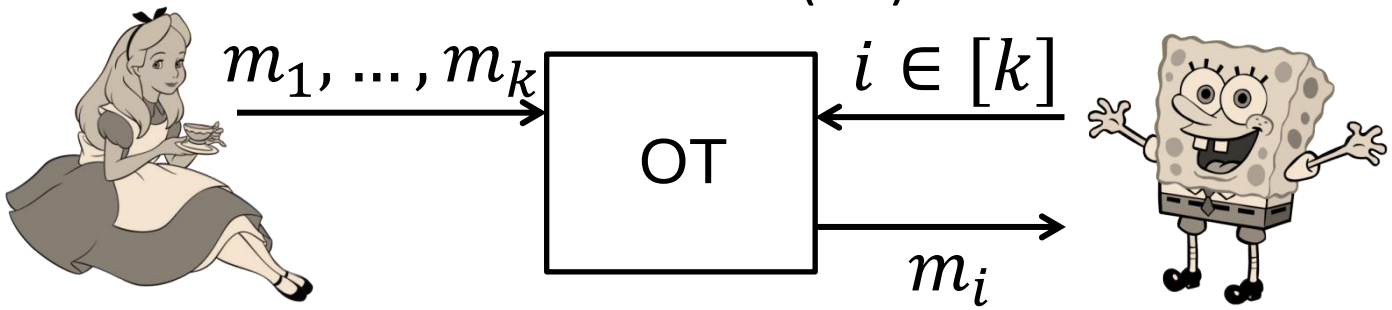
$\mathcal{F}_B := (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

For $\mathcal{F}_B \setminus \mathcal{F}_A$ compute $f_i(x)$

$$m_i := \begin{cases} f_i(x) & \text{if } f_i \notin \mathcal{F}_A \\ \perp & \text{otherwise} \end{cases}$$

Naive OT-based Idea to Realise rCPFE

- A tool: 1 out of k Oblivious Transfer (OT):



- A naive rCPFE protocol:

$x, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B$

$\mathcal{F}_B := (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

For $\mathcal{F}_B \setminus \mathcal{F}_A$ compute $f_i(x)$

$$m_i := \begin{cases} f_i(x) & \text{if } f_i \notin \mathcal{F}_A \\ \perp & \text{otherwise} \end{cases}$$

$m_1, \dots, m_k$

OT

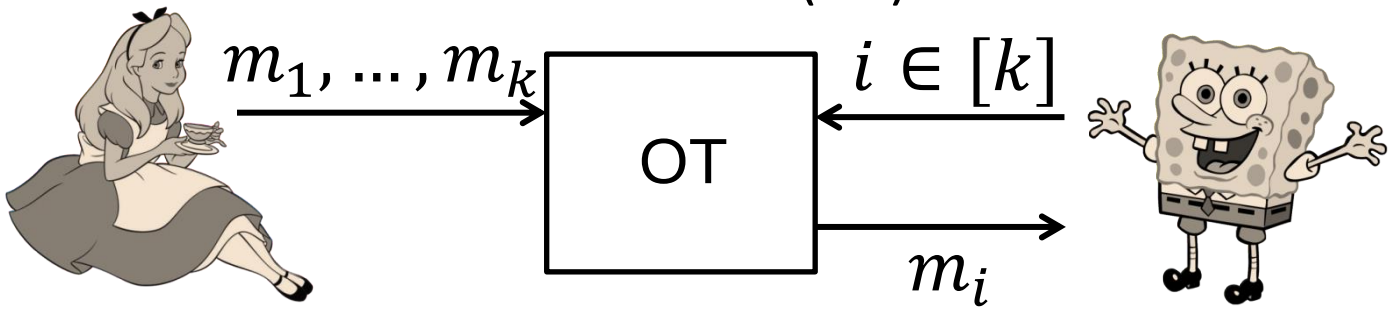
$i+1$

m_{i+1}

$m_{i+1} = f(x)$

Naive OT-based Idea to Realise rCPFE

- A tool: 1 out of k Oblivious Transfer (OT):



- A naive rCPFE protocol:

$x, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B \coloneqq (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

$\mathcal{F}_B$

For $\mathcal{F}_B \setminus \mathcal{F}_A$ compute $f_i(x)$

$$m_i \coloneqq \begin{cases} f_i(x) & \text{if } f_i \notin \mathcal{F}_A \\ \perp & \text{otherwise} \end{cases}$$

$i+1$

$m_1, \dots, m_k$

OT

m_{i+1}

$m_{i+1} = f(x)$

Another Tool: Functional Encryption (FE)

- Recall traditional (secret key) encryption!

$Setup(\lambda) \rightarrow msk$

$Enc(msk, x) \rightarrow CT_x$

$Dec(CT_x, msk) \rightarrow x$

Another Tool: Functional Encryption (FE)

- Recall traditional (secret key) encryption!

$FE.Setup(\lambda) \rightarrow msk$

$FE.Enc(msk, x) \rightarrow CT_x$

$\boldsymbol{FE.KeyDer(msk, f) \rightarrow sk_f}$

$FE.Dec(CT_x, \boldsymbol{sk_f}) \rightarrow \boldsymbol{f(x)}$

- FE generalizes traditional encryption [BSW11]
- Decryption reveals about x no more than $f(x)$
- Arbitrary keys and ciphertexts can be combined in decryption
- To good to be true?
  - For **general** functions: impossibility/use of untested assumptions [BSW11]
  - For **restricted** functions: secure consructions, e.g. [GVW12, ALS16]

Generic rCPFE based on FE & OT

$x_1, \ldots, x_d, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

ONLINE PHASE

Generic rCPFE based on FE & OT

$x_1, \dots, x_d, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

ONLINE PHASE

Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B$ ←

$\mathcal{F}_B := (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

Generic rCPFE based on FE & OT

$x_1, \dots, x_d, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

ONLINE PHASE

Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B \coloneqq (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

$\mathcal{F}_B$ ←

Runs $FE.Setup(\lambda) \rightarrow msk$

$\forall i \in [d]$: $FE.Enc(msk, x_i) = CT_{x_i}$

$CT_{x_1}, \dots, CT_{x_d}$ →

Generic rCPFE based on FE & OT

$x_1, \dots, x_d, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

ONLINE PHASE

Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B \coloneqq (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

$\mathcal{F}_B$ ←

Runs $FE.Setup(\lambda) \rightarrow msk$

$\forall i \in [d]$: $FE.Enc(msk, x_i) = CT_{x_i}$

$CT_{x_1}, \dots, CT_{x_d}$ →

For $\mathcal{F}_B \setminus \mathcal{F}_A$ compute:

$FE.KeyDer(msk, f_j) = sk_{f_j}$

$$m_j \coloneqq \begin{cases} sk_{f_j} \text{ if } f_j \notin \mathcal{F}_A \\ \perp \text{ otherwise} \end{cases}$$

Generic rCPFE based on FE & OT

$x_1, \dots, x_d, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

ONLINE PHASE

Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B \coloneqq (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

$\mathcal{F}_B$ ←

Runs $FE.Setup(\lambda) \to msk$

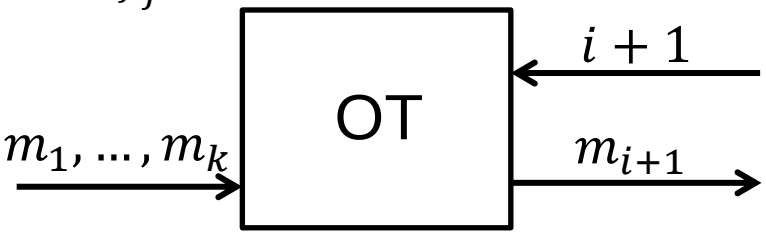
$\forall i \in [d]$: $FE.Enc(msk, x_i) = CT_{x_i}$

$CT_{x_1}, \dots, CT_{x_d}$ →

For $\mathcal{F}_B \setminus \mathcal{F}_A$ compute:

$FE.KeyDer(msk, f_j) = sk_{f_j}$

$$m_j \coloneqq \begin{cases} sk_{f_j} \text{ if } f_j \notin \mathcal{F}_A \\ \perp \text{ otherwise} \end{cases}$$



$m_{i+1} = sk_f$

Generic rCPFE based on FE & OT

$x_1, \dots, x_d, \mathcal{F}_A$ *INPUTS* $f \in \mathcal{F}$

ONLINE PHASE

Sample $f_i \in \mathcal{F}$ for $i \in [k-1]$

$\mathcal{F}_B \coloneqq (f_1, \dots, f_i, f, f_{i+1}, \dots, f_{k-1})$

$\mathcal{F}_B$ ←

Runs $FE.Setup(\lambda) \to msk$

$\forall i \in [d]$: $FE.Enc(msk, x_i) = CT_{x_i}$

$CT_{x_1}, \dots, CT_{x_d}$ →

For $\mathcal{F}_B \setminus \mathcal{F}_A$ compute:

$FE.KeyDer(msk, f_j) = sk_{f_j}$

$$m_j \coloneqq \begin{cases} sk_{f_j} \text{ if } f_j \notin \mathcal{F}_A \\ \perp \text{ otherwise} \end{cases}$$

$m_1, \dots, m_k$ →

OT

$i+1$ ←

m_{i+1} →

$m_{i+1} = sk_f$

OFFLINE PHASE

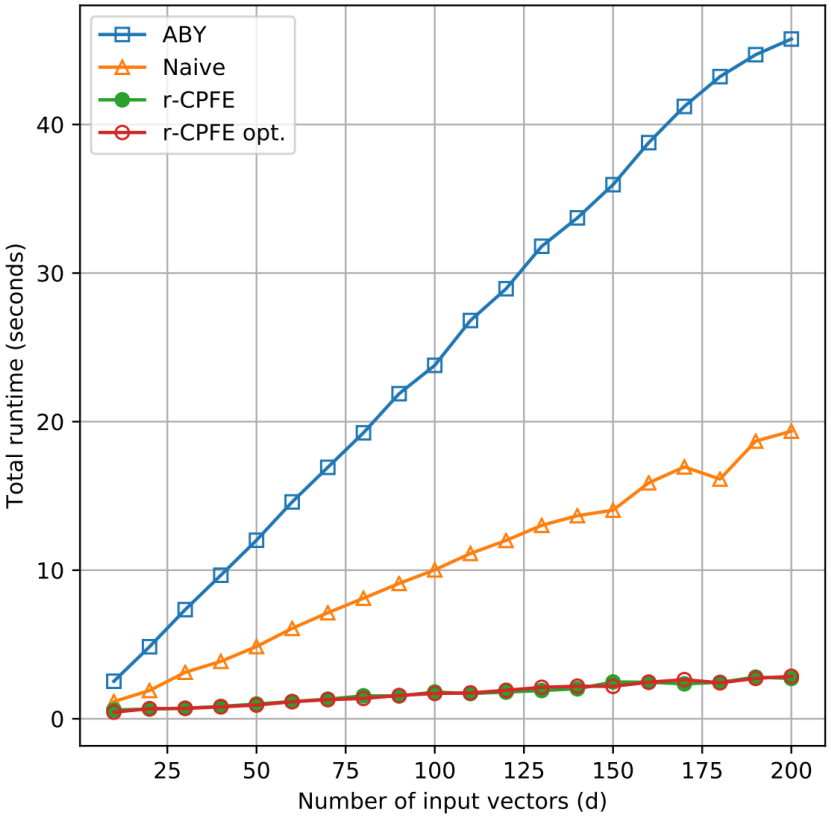
$\forall i \in [d]$: $FE.Dec(CT_{x_i}, sk_f) \to f(x_i)$

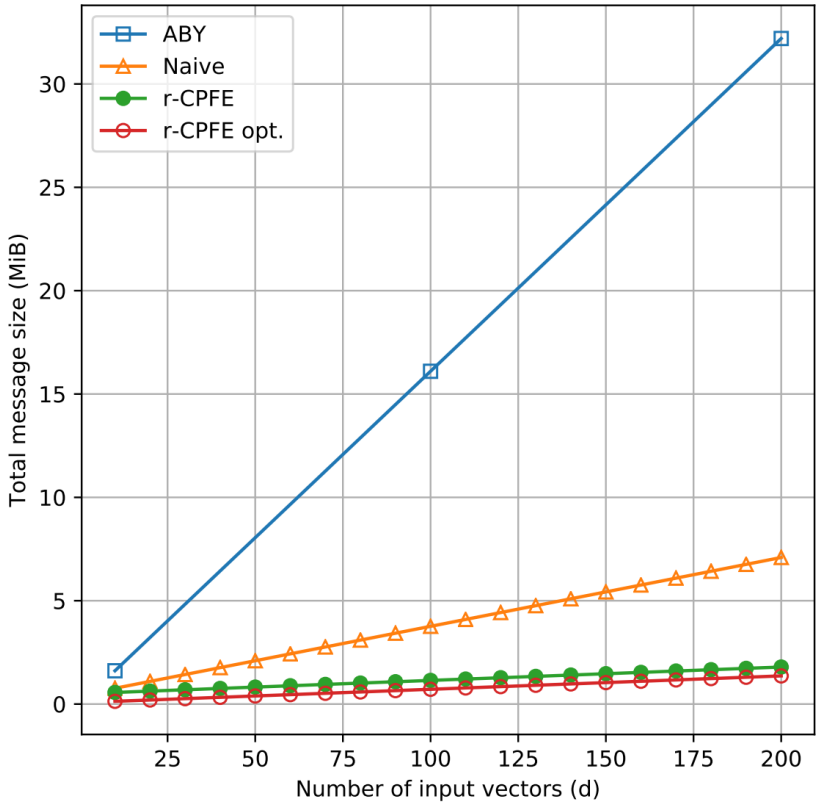
Generic rCPFE based on FE & OT

- Theorem. /informal/
  The protocol is SIM secure against semi-honest adversaries, if the underlying *FE scheme is k-query non-adaptive SIM secure for a single message* and the used *OT protocol is SIM secure against semi-honest adversaries*.
- Corollary.
  The protocol achieves *strong relaxed function privacy* if all dummy functions are sampled from the same distribution as f and $aux = \perp$.
- Instantiations:
  - Using FE of [GVW12] → rCPFE for $P/poly$ (in theory!)
  - Using inner product FE [ALS16] → inner product rCPFE from the DDH assumption
    Application: statistical analysis, e.g. logistic regression, etc.

Performance of Our Inner Product rCPFE

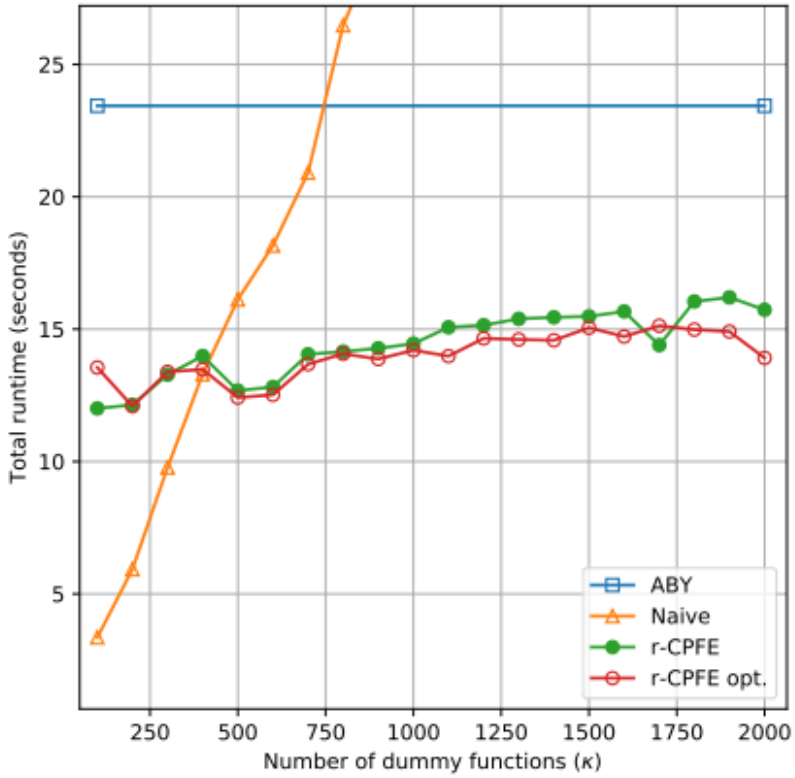
- Comparison with the state of the art secure inner product protocol (ABY framework) [DSZ15]:
  - Vector dimension: 100
  - Number of dummy functions: 1000

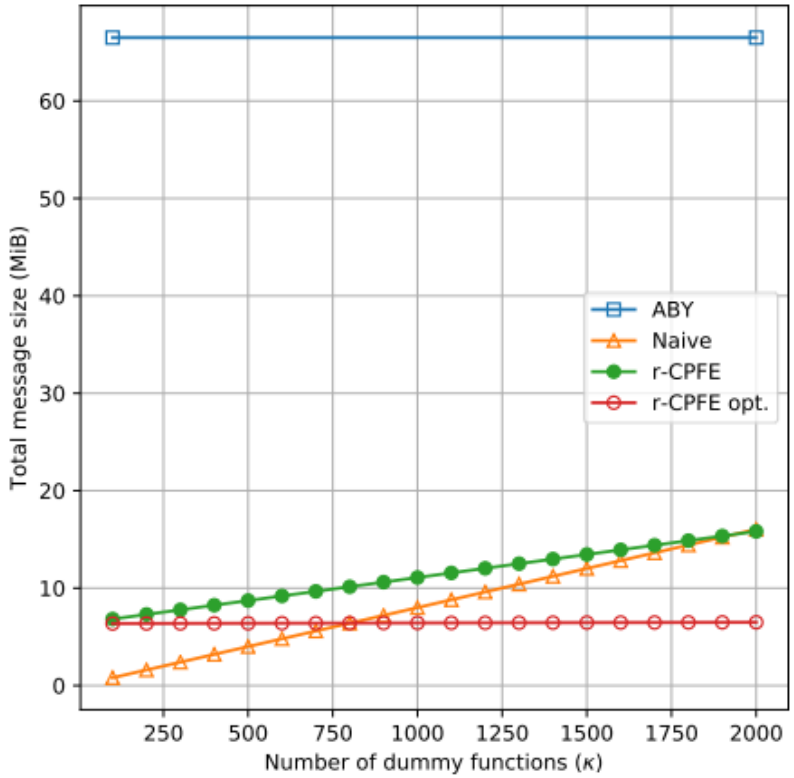




Performance of Our Inner Product rCPFE

- Representating the cost of our dummy functions
  - Vector dimension: 1000
  - Number evaluations with different inputs: 100
  - Approx equality of running time (with ABY) is for k=6200





Summary & Open Directions

- We initiate the study of partiol information leakage in PFE
- Defined variants of CPFE
- Generic realizations
  - Focus on scenarios where multiple evaluations occur
  - Trade-off to achieve better performance
- Inner Product rCPFE is practical

Future work

- Non-zero auxiliary info?
- How to get rid of dummy functions?
- Different trade-offs?

Take-home message

- Parital information can be sensitive and can leak easily!

MŰEGYETEM 1782

www.crysys.hu

Thank you for the attention!

Q&A

References

[ALS16] Agrawal, S., Libert, B., Stehlé, D.*: Fully secure functional encryption for innerproducts, from standard assumptions.* In: Robshaw, M., Katz, J. (eds.) Advancesin Cryptology - CRYPTO 2016,

[BSW11] Boneh, Dan, Amit Sahai, and Brent Waters. ***Functional encryption: Definitions and challenges***. Theory of Cryptography Conference. Springer, Berlin, Heidelberg, 2011.

[DSZ15] Demmler, D., Schneider, T., Zohner, M.: ***ABY - A framework for efficient mixed-protocol secure two-party computation***. In: 22nd AnnualNetwork and Distributed System Security Symposium, NDSS 2015.

[GVW12] Gorbunov, S., Vaikuntanathan, V., Wee, H.: ***Functional encryption with bounded collusions via multi-party computation***. In: Safavi-Naini, R., Canetti, R. (eds.)Advances in Cryptology - CRYPTO 2012.

[PSS09] Paus, A., Sadeghi, A. R., & Schneider, T. (2009, June). ***Practical secure evaluation of semi-private functions***. ACNS-2009