EUROCRYPT 2024

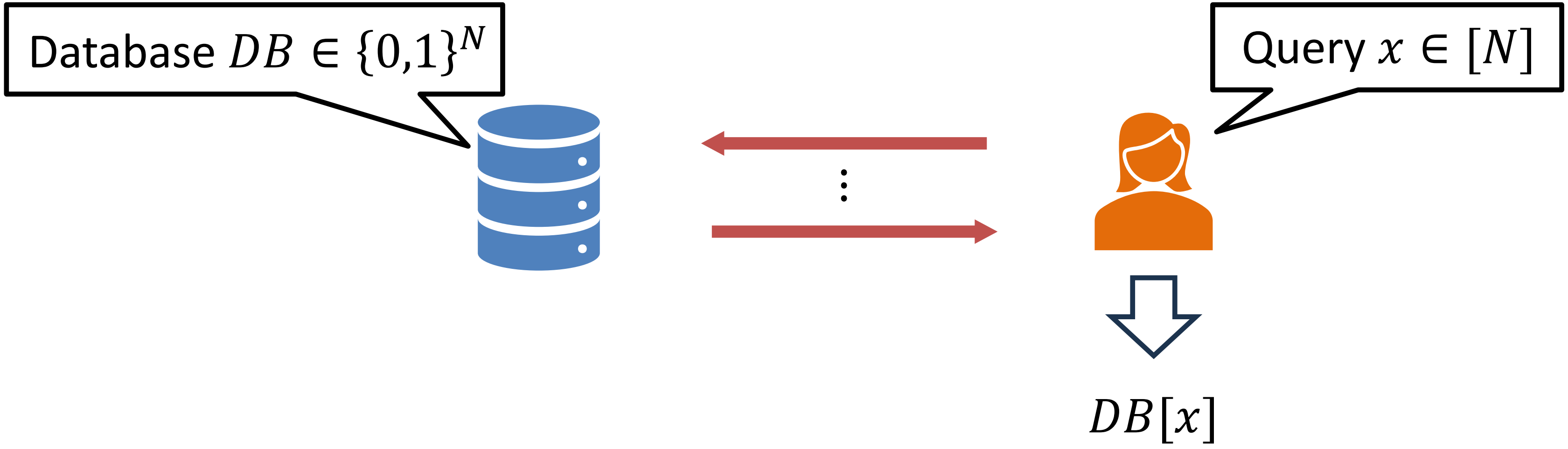
Efficient and Generic Methods to Achieve Active Security in Private Information Retrieval and More Advanced Database Search

May 29, 2024

Reo Eriguchi,[1] Kaoru Kurosawa,[2,1] Koji Nuida[3,1]

1. AIST, Japan
2. Chuo University, Japan
3. Kyushu University, Japan

Private Information Retrieval



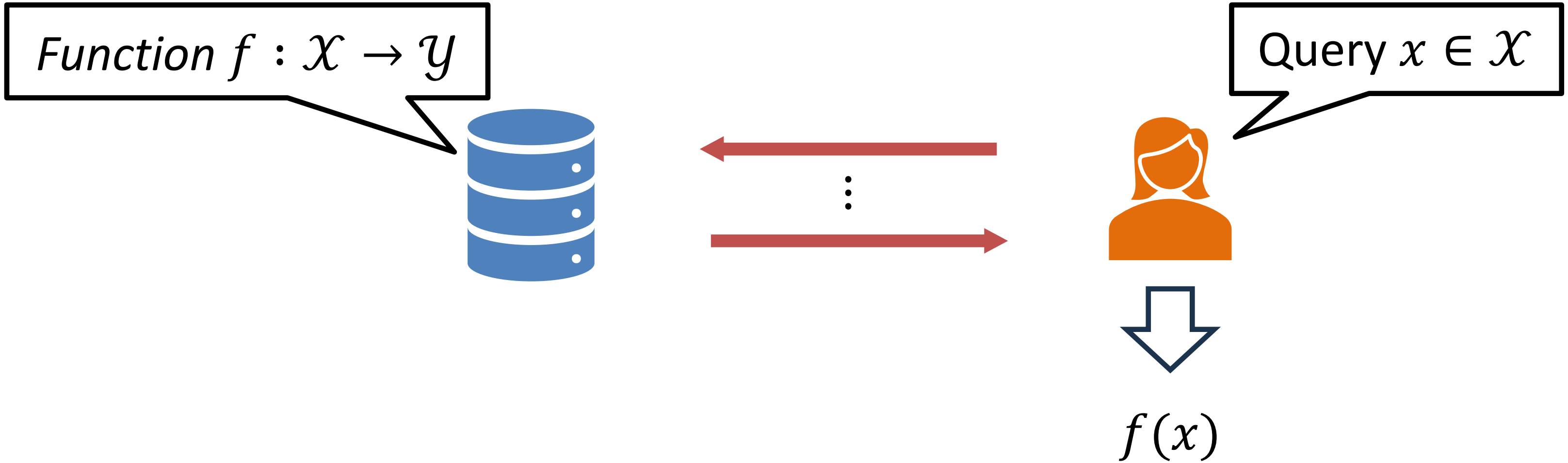
Privacy

Servers learn no information on the client's query x.

The privacy of DB is not considered.

Secure Database Search

- We consider a more general setting of computing a *function* f.

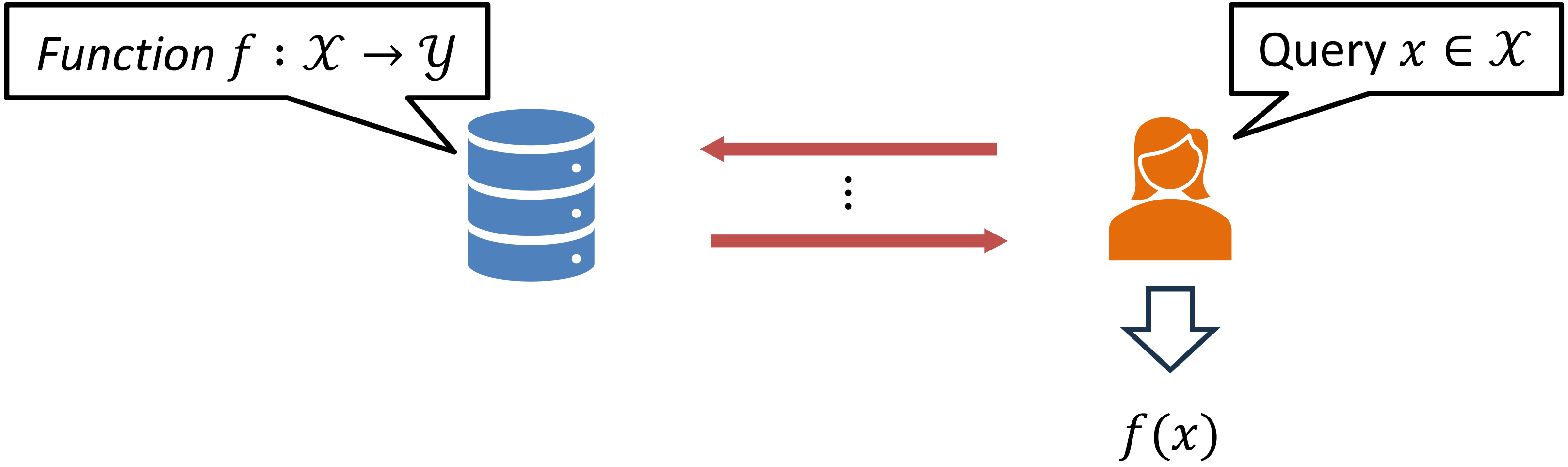


Privacy

Servers learn no information on the client's query x.
The privacy of f is not considered.

Secure Database Search

- We consider a more general setting of computing a *function* f.



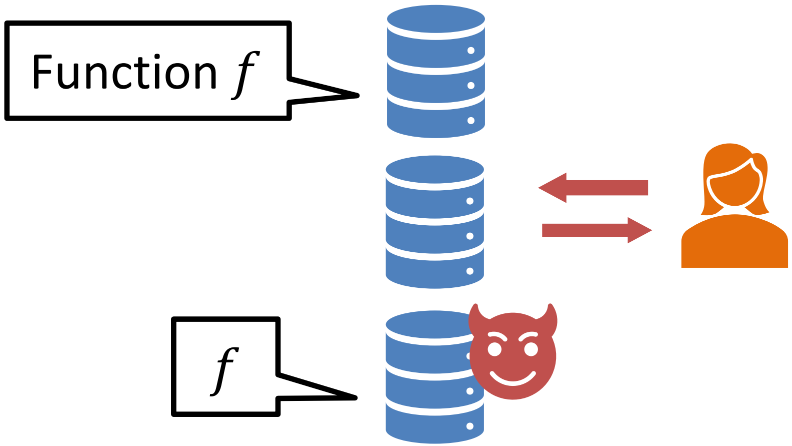
Trivial solution Client can download f and compute $f(x)$ locally.

However, Client-side computation/communication is proportional to $|f|$.

Question Protocols whose computation/communication is $\ll |f|$.

Multi-server vs Single-server

Multi-server setting



- ✔ Better efficiency
- ✔ Weaker assumption
- ✘ # of corrupted servers $\leq t$

This work

Single-server setting

- ✘ Heavy computation
- ✘ Stronger assumption

(Unconditional security cannot be achieved)

Passively Secure Protocols

- **Passive t-security**: Semi-honest adversary corrupts t servers.
- Private information retrieval (PIR)

  2^t-server protocol from OWF [BGI16]+[BIW10],

  3^t-server protocol (unconditional) [BIKO10]+[BIW10],…

- Degree-D polynomial

  $\Theta(tD)$-server protocol [WY07],

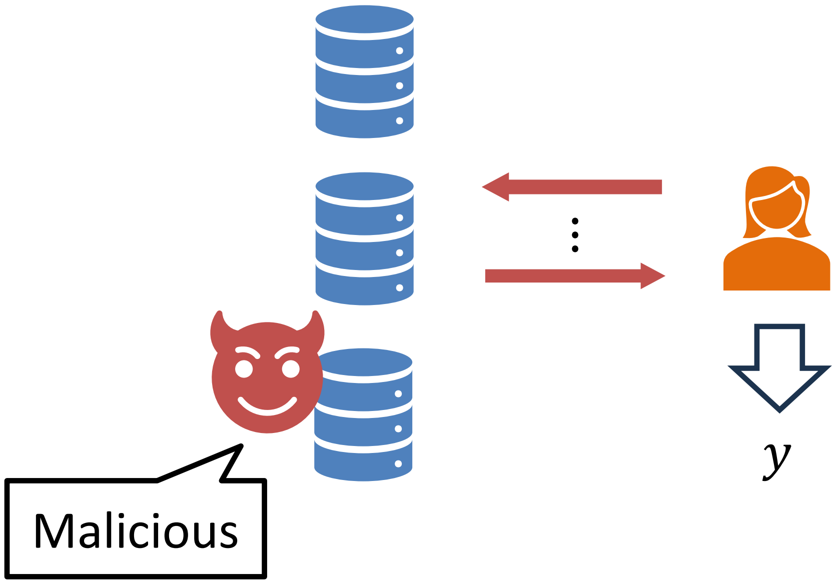
  $(t+1)$-server protocol for $D = o(\log \lambda)$ from sparse LPN [DIJL23]

- Constant-depth circuits of size M

  $(t \cdot \mathrm{polylog}\, M)$-server protocol [BI05]

Active Security

- Corrupted servers may deviate from a protocol.



<u>Privacy</u>

Corrupted servers learn no information on x.

<u>Byzantine-robustness</u>

$y = f(x)$ with high probability.

cf. <u>Verifiability [CNC+23,ZW22]</u>

$y \in \{f(x), \perp\}$ with high probability.

Previous Works

- Passive-to-active compilers were proposed for *PIR* [BS07], [EKN22].

1-round **passively t-secure** k-server protocol → **Actively t-secure** m-server protocol

Compiler	# servers	# rounds	Function
[BS07]	$m = k + 2t$	1	PIR
[EKN22]	$m = k + t$	1	PIR

✘ $\binom{m}{t} = m^{O(t)}$ computation/communication overhead.

✘ Do not consider general functions.

Our Results

- We propose **generic** passive-to-active compilers with **polynomial overheads**.

Compiler	# servers	# rounds	Function
[BS07]	$m = k + 2t$	1	PIR
[EKN22]	$m = k + t$	1	PIR
Ours-1	$m = k + t$	$O(m^2)$	Any
Ours-2	$m = \Theta(k \log k) + 2t$	1	Any

✔ General f

✔ $\mathrm{poly}(m)$ computation/communication overhead

Techniques of Our Compilers

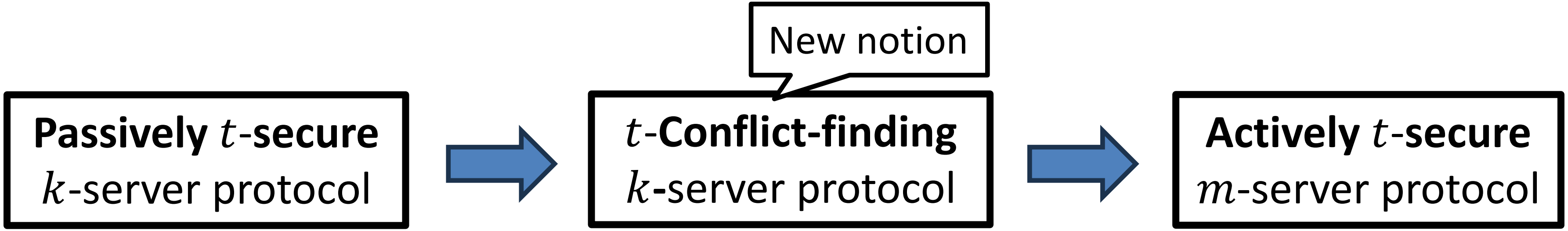
1-round
passively t-secure
k-server protocol

→

Actively t-secure
m-server protocol

Compiler	# servers	# rounds	Function
[BS07]	$m = k + 2t$	1	PIR
[EKN22]	$m = k + t$	1	PIR
Ours-1	$m = k + t$	$O(m^2)$	Any
Ours-2	$m = \Theta(k \log k) + 2t$	1	Any

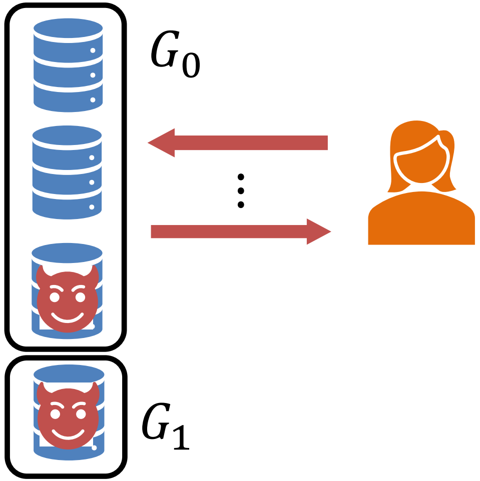
Conflict-finding Protocol



Conflict-finding Protocol

New notion

Passively t-secure k-server protocol → **t-Conflict-finding** k-server protocol → **Actively t-secure** m-server protocol



Privacy Same

Conflict-finding

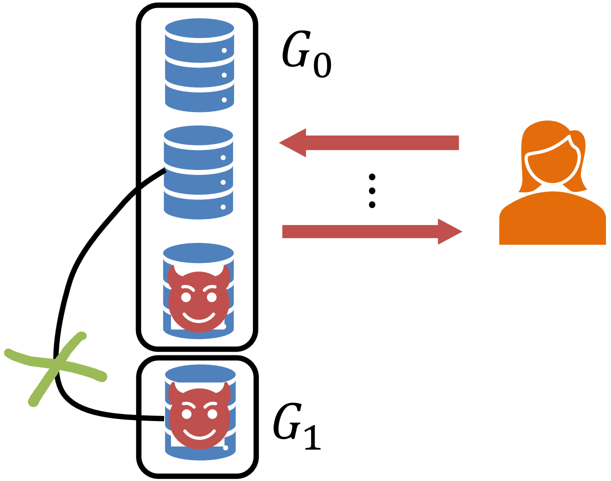
Client outputs either
(1) $f(x)$, or
(2) a partition (G_0, G_1) such that $H \subseteq G_0$ or $H \subseteq G_1$.

Set of honest servers

Conflict-finding Protocol

New notion

Passively t-secure k-server protocol → **t-Conflict-finding** k-server protocol → **Actively t-secure** m-server protocol



Privacy Same

Conflict-finding

Client outputs either
(1) $f(x)$, or
(2) a partition (G_0, G_1) such that $H \subseteq G_0$ or $H \subseteq G_1$.

Set of honest servers

Conflicting pair

For any pair (S_i, S_j) with $S_i \in G_0, S_j \in G_1$, we have that S_i or S_j is malicious.

Conflict-finding Protocol

New notion

Passively t-secure k-server protocol → t-Conflict-finding k-server protocol → Actively t-secure m-server protocol

Privacy Same

Conflict-finding

Client outputs either
(1) $f(x)$, or
(2) a partition (G_0, G_1) such that $H \subseteq G_0$ or $H \subseteq G_1$.

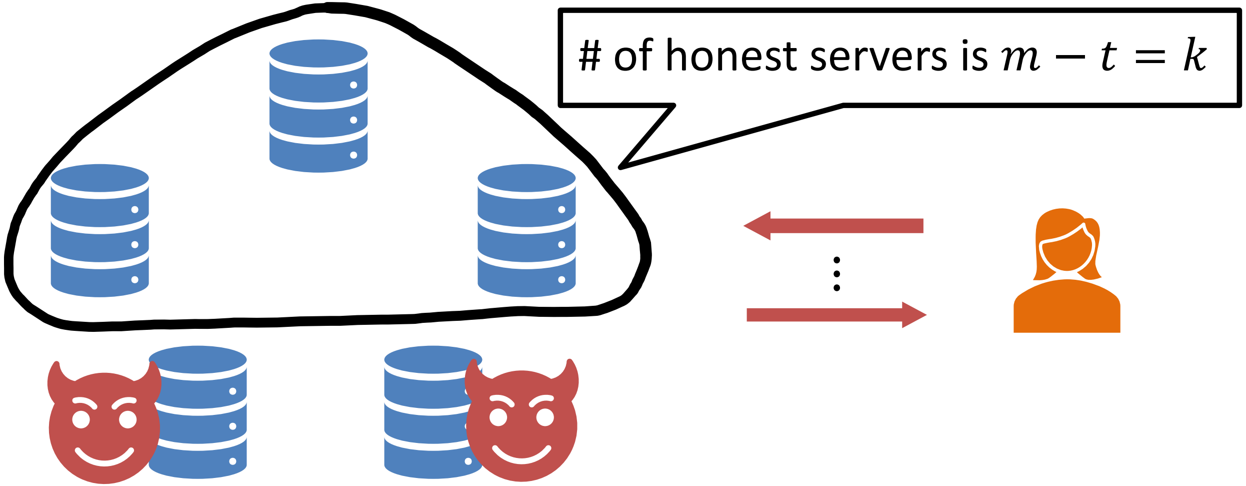
Set of honest servers

Conflicting pair

For any pair (S_i, S_j) with $S_i \in G_0, S_j \in G_1$, we have that S_i or S_j is malicious.

From Conflict-finding to Actively Secure

- There are m servers out of which t are malicious.



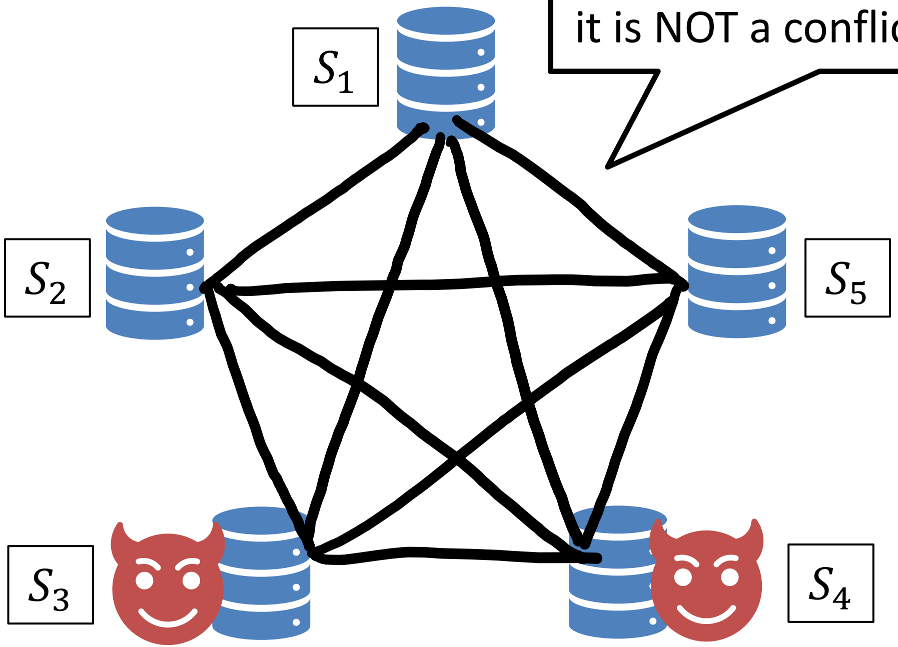
- If a k-server protocol is executed with the set of honest servers,
  Client obtains a correct result $f(x)$.

Strategy | Find sufficiently many conflicting pairs to determine k honest servers

From Conflict-finding to Actively Secure

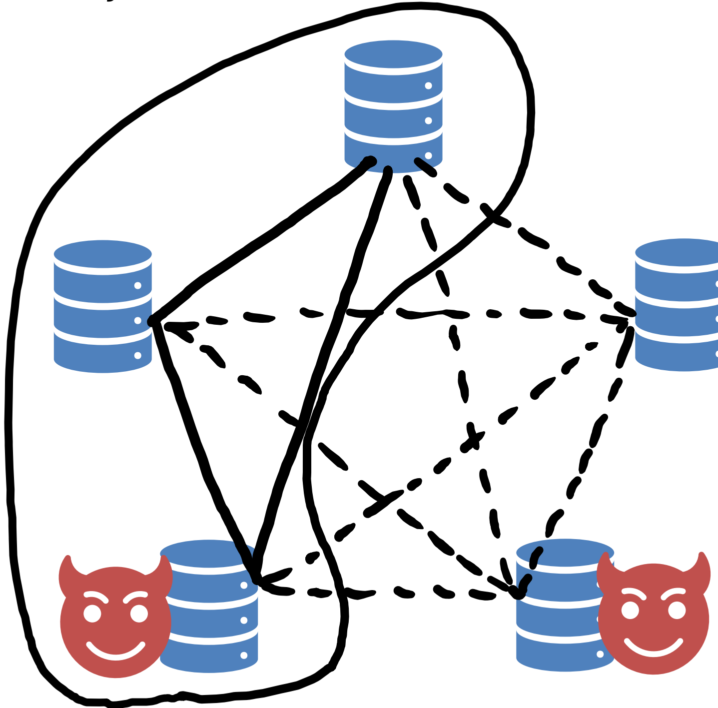
- We consider a graph whose nodes represent servers.
  - An initial graph is a complete graph.

(S_i, S_j) is an edge if and only if it is NOT a conflicting pair.



From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.

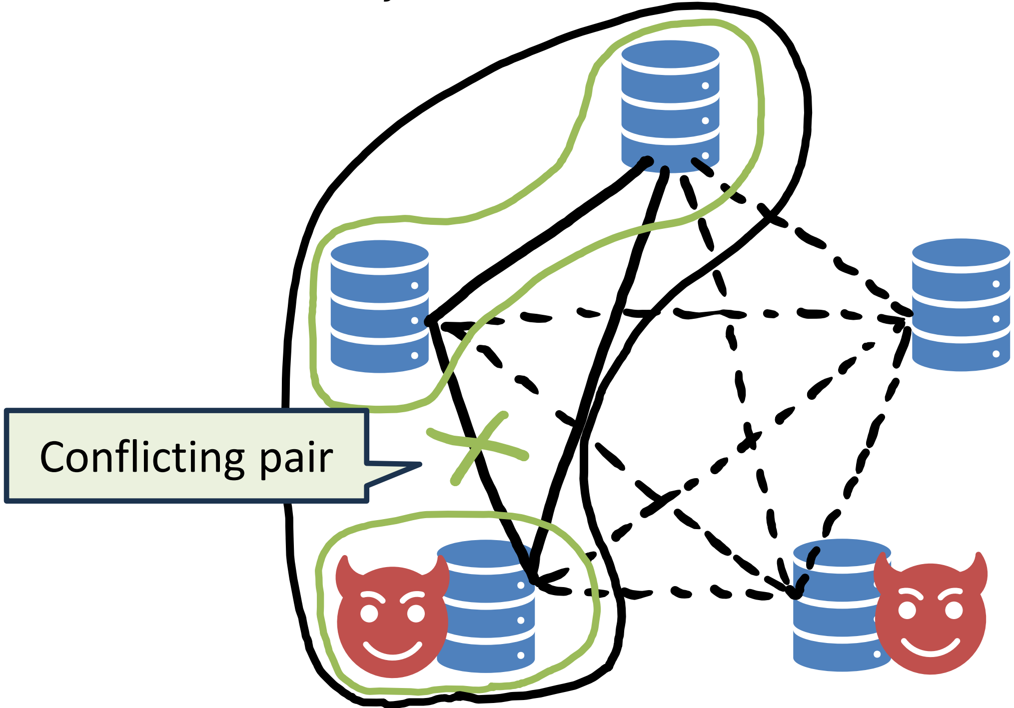


From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.

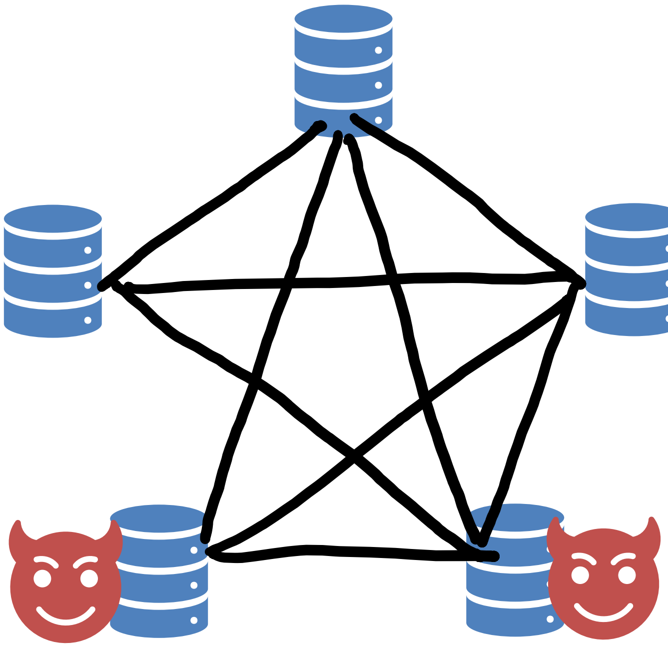
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



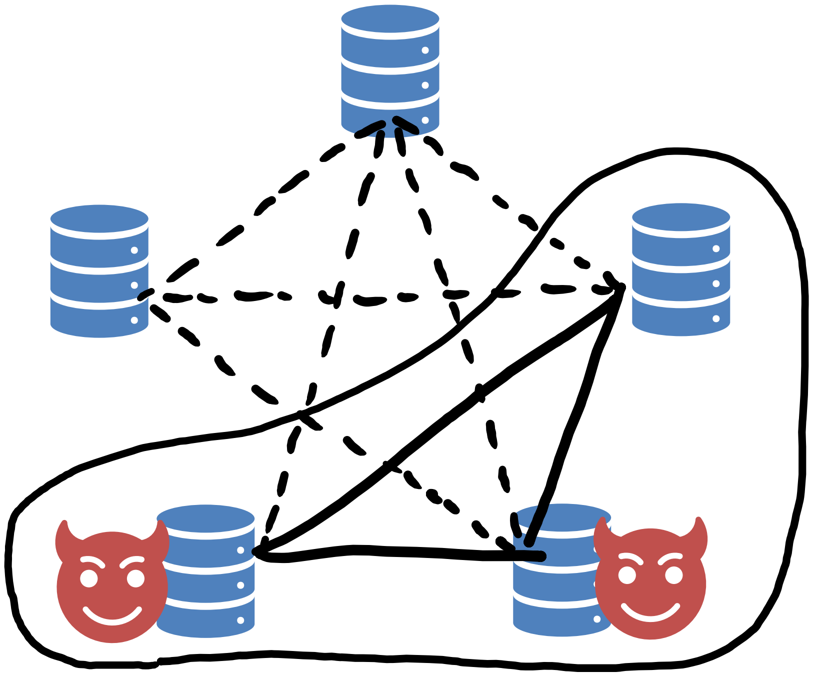
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



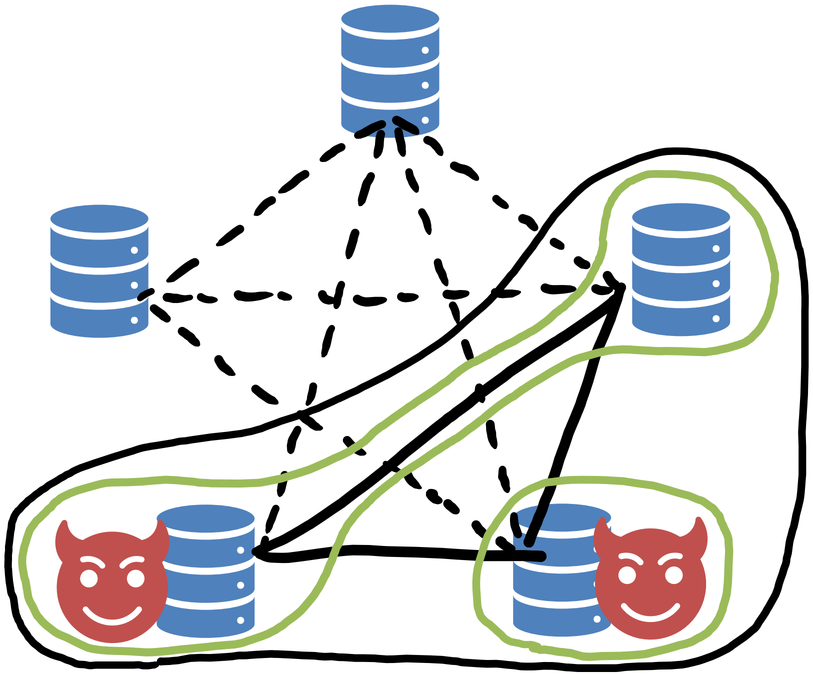
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



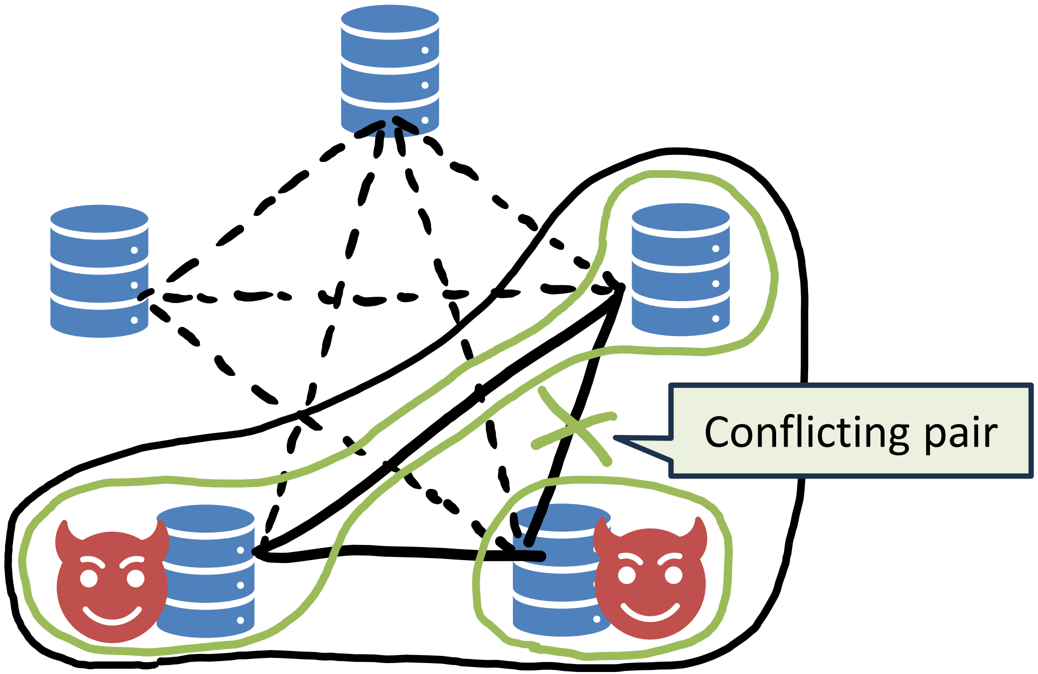
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



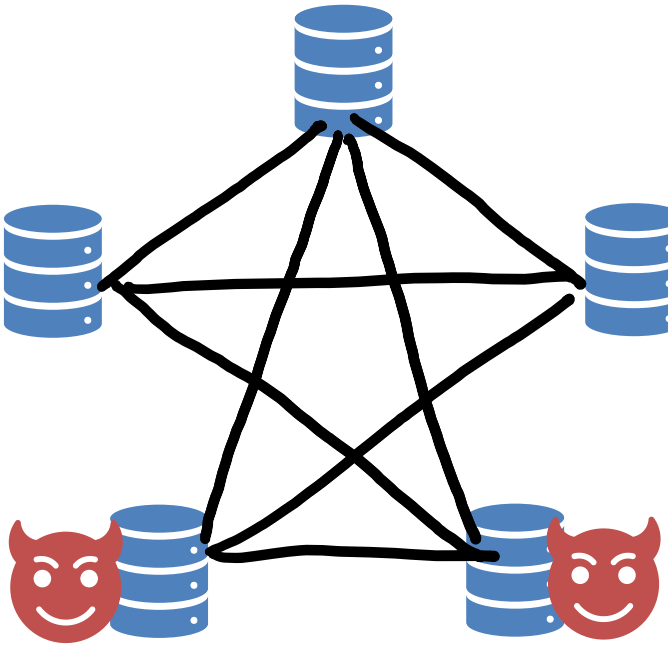
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



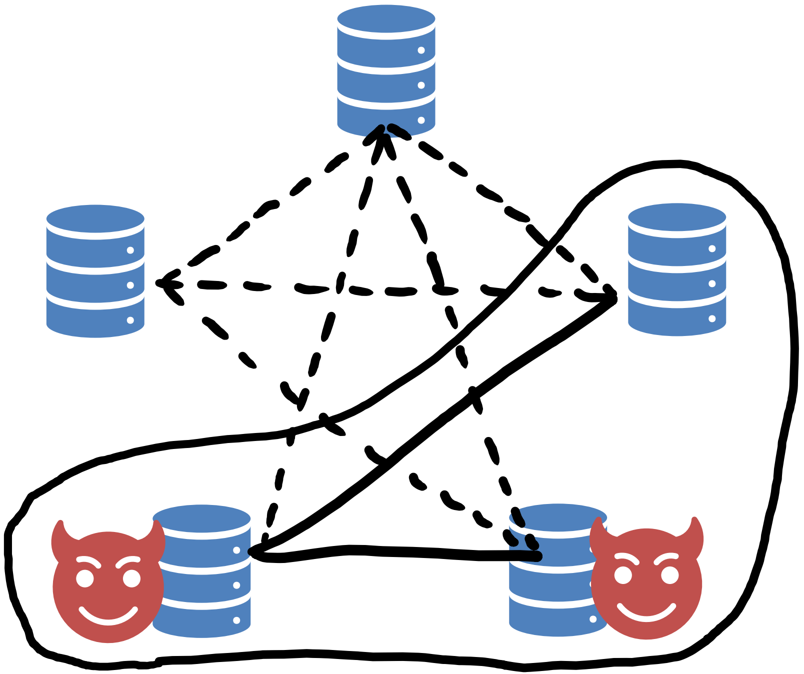
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



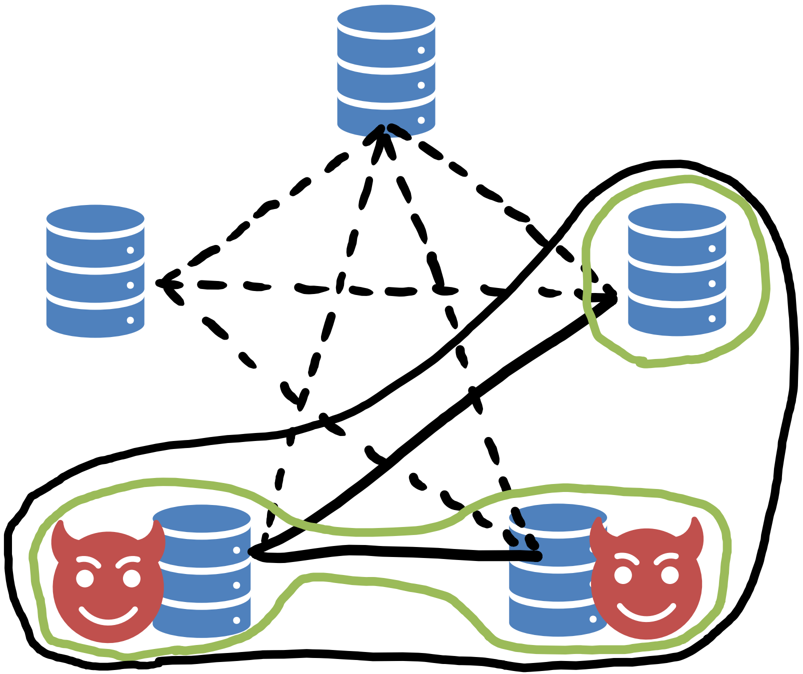
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



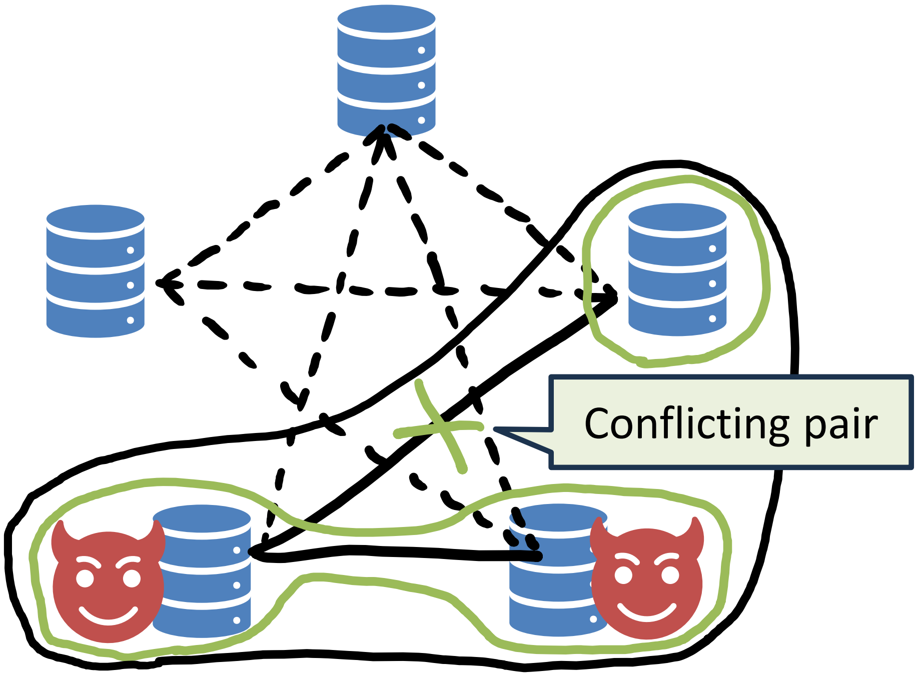
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



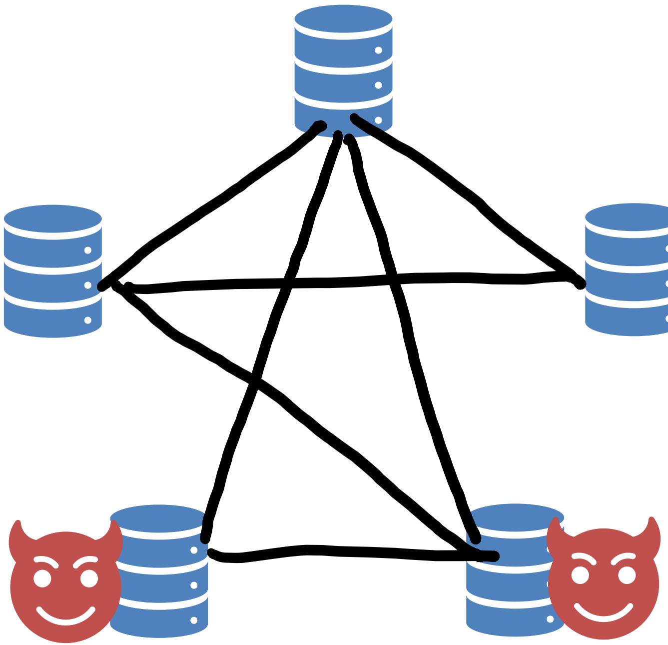
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



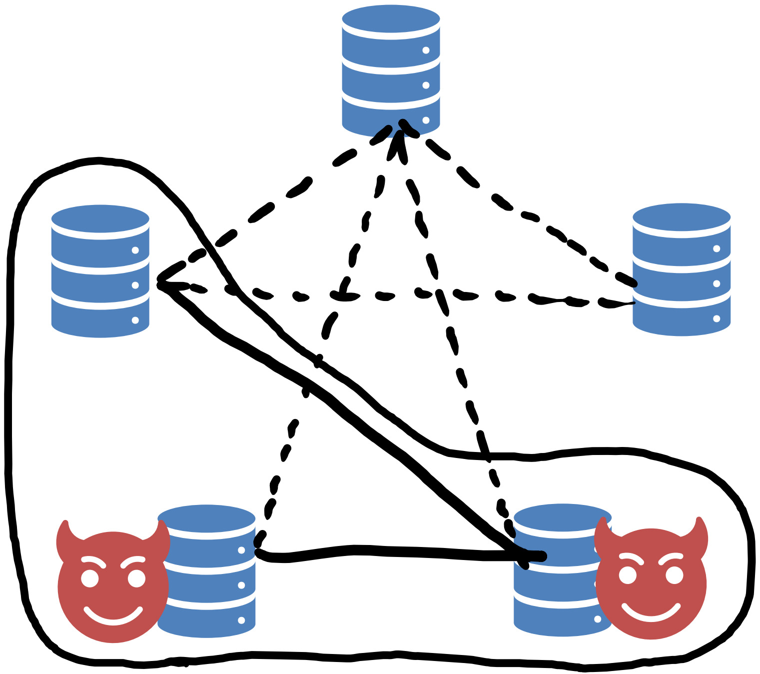
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



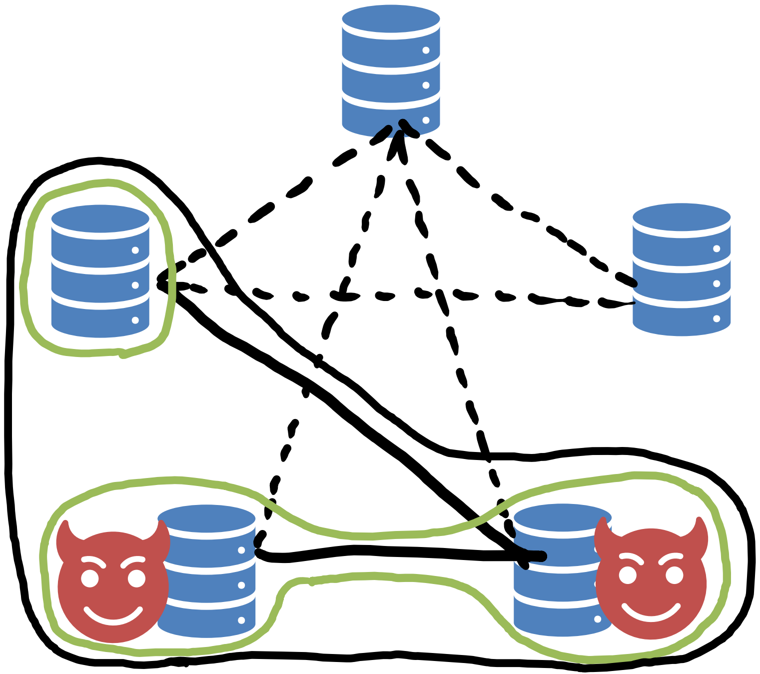
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



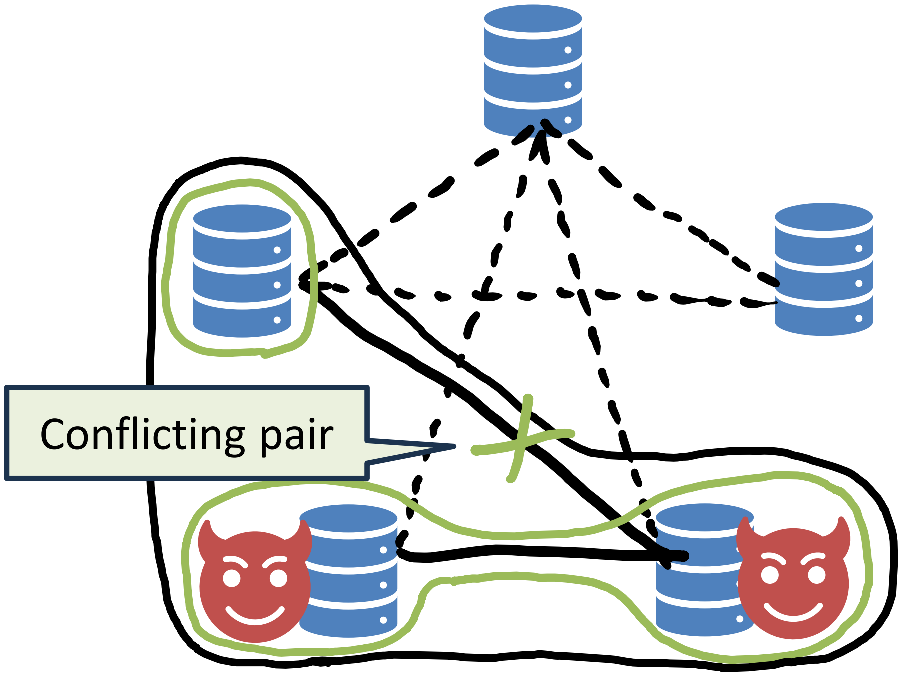
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



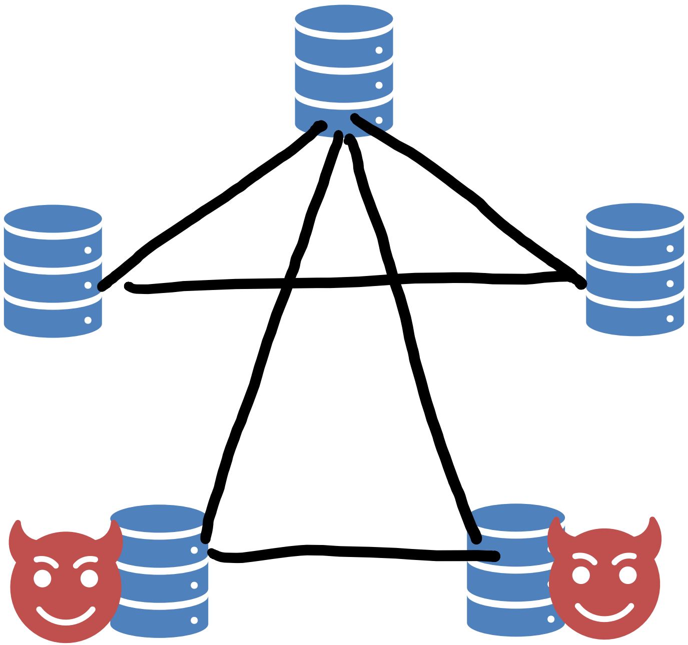
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



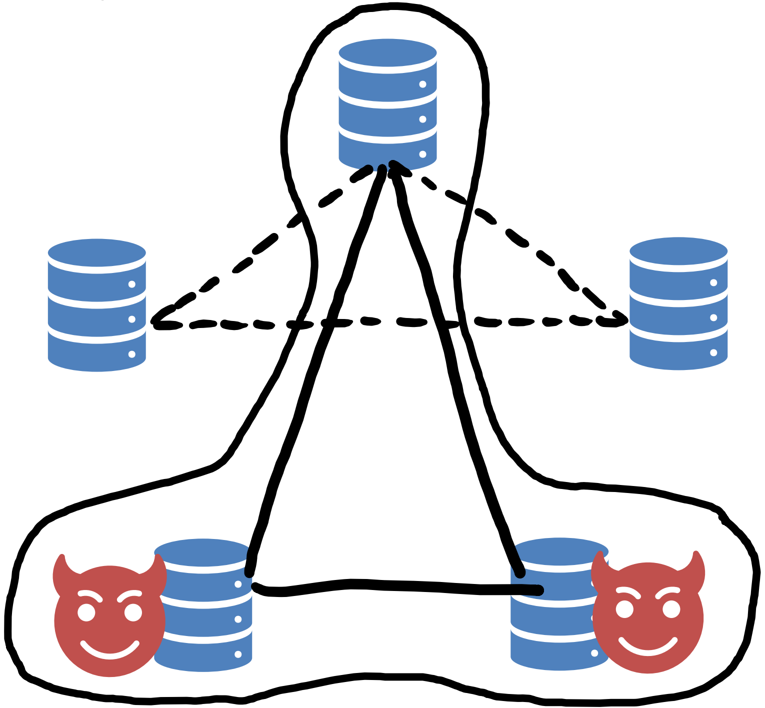
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



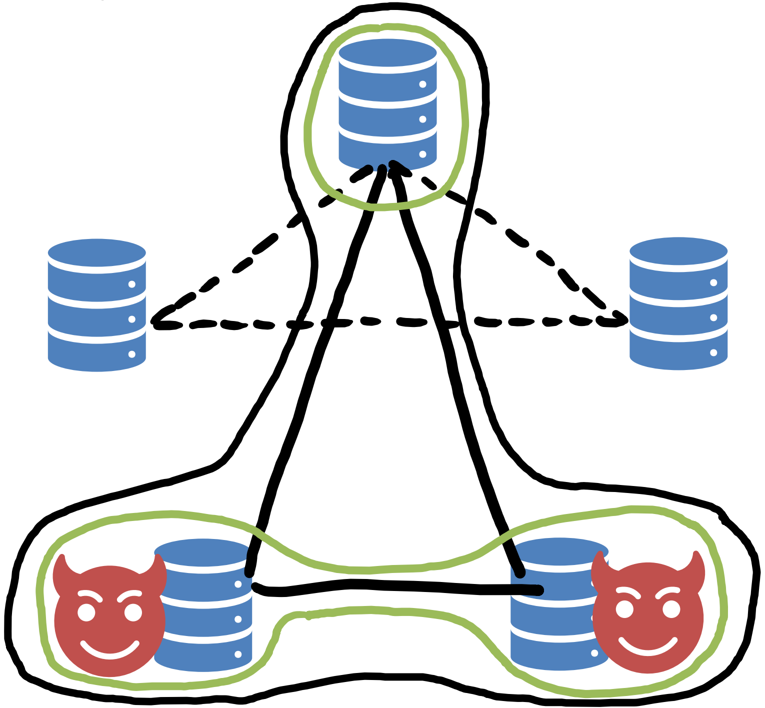
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



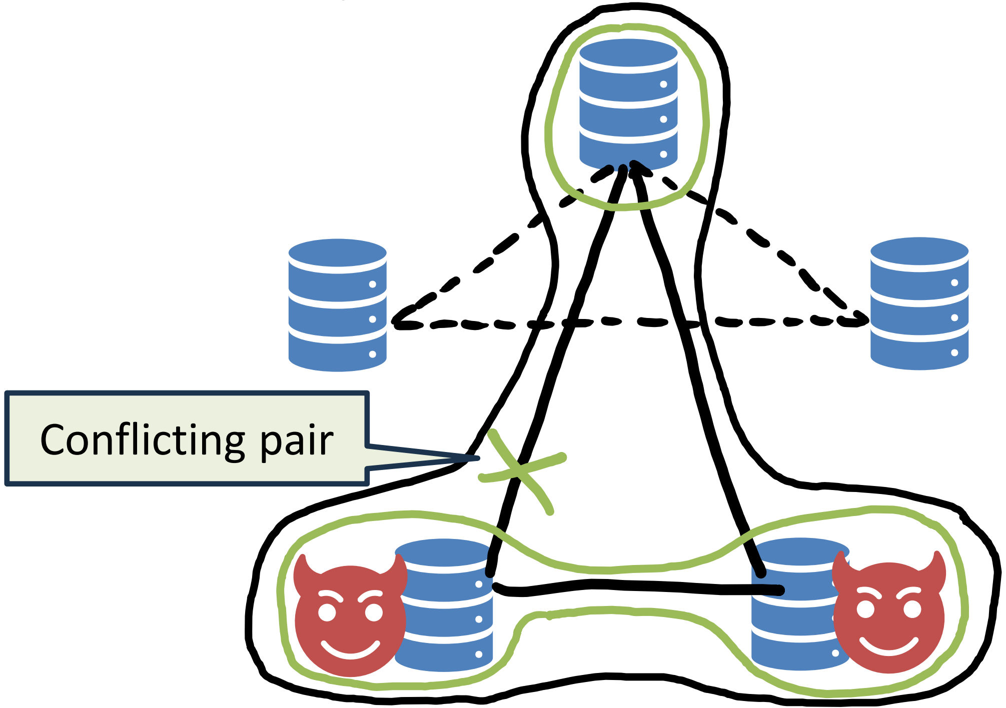
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



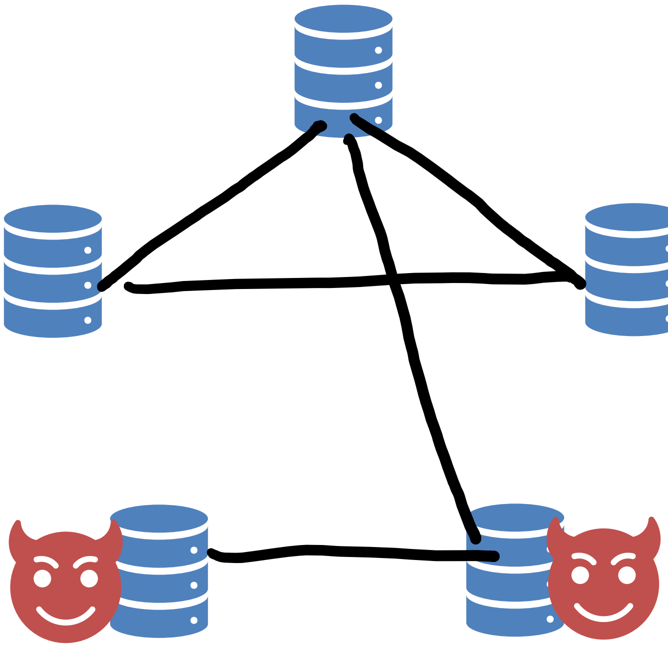
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



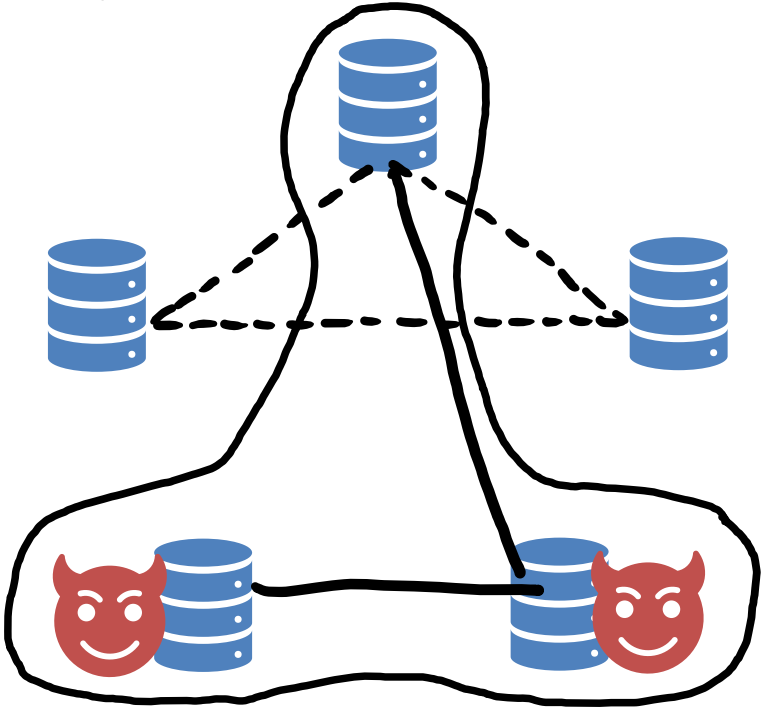
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



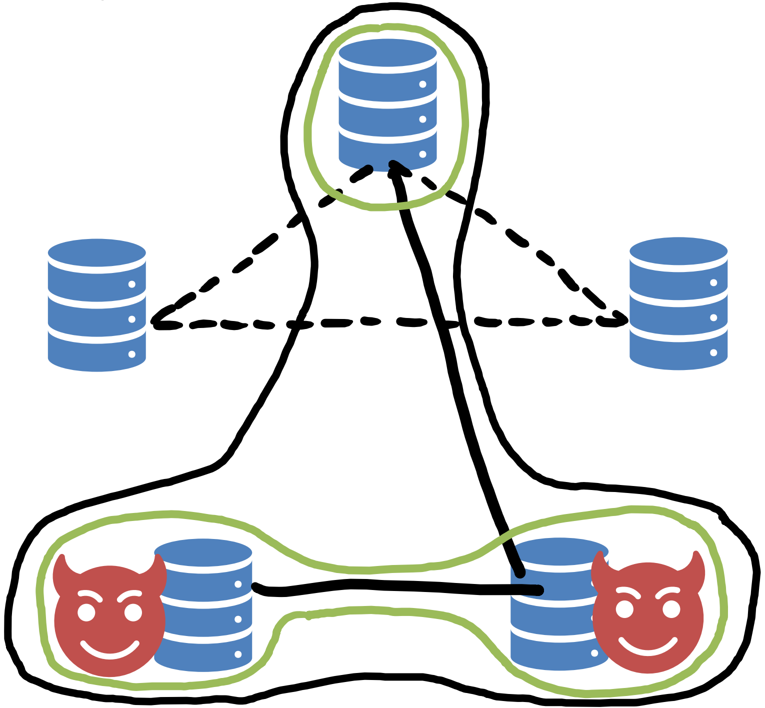
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



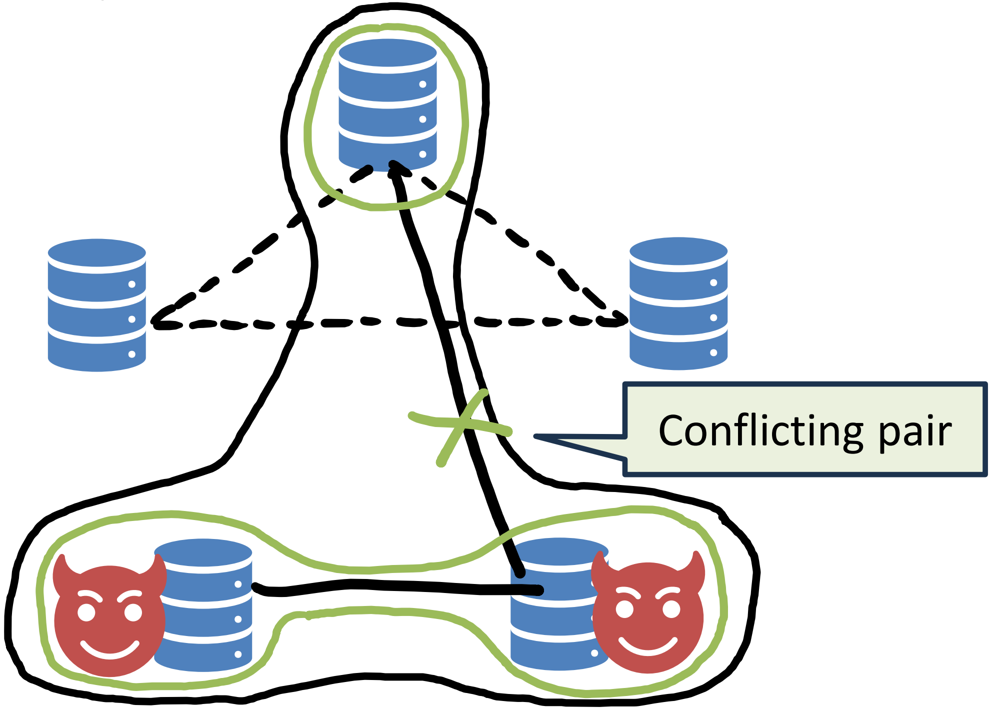
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



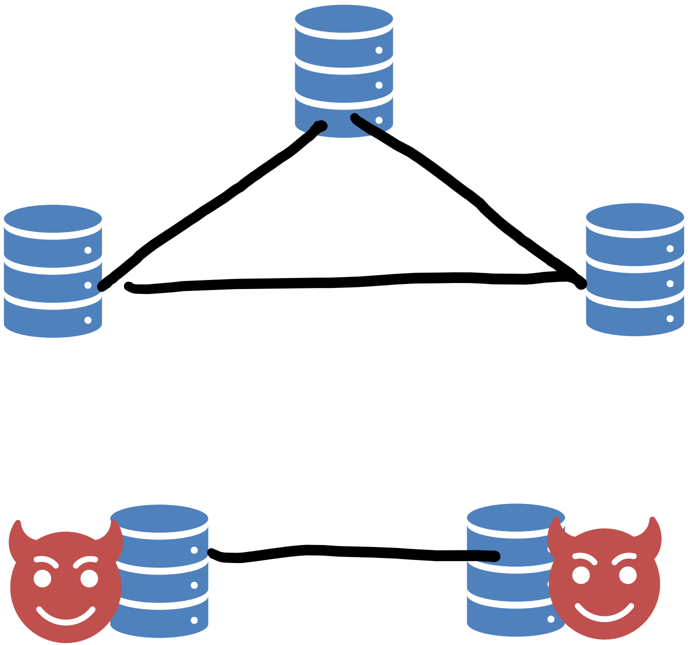
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



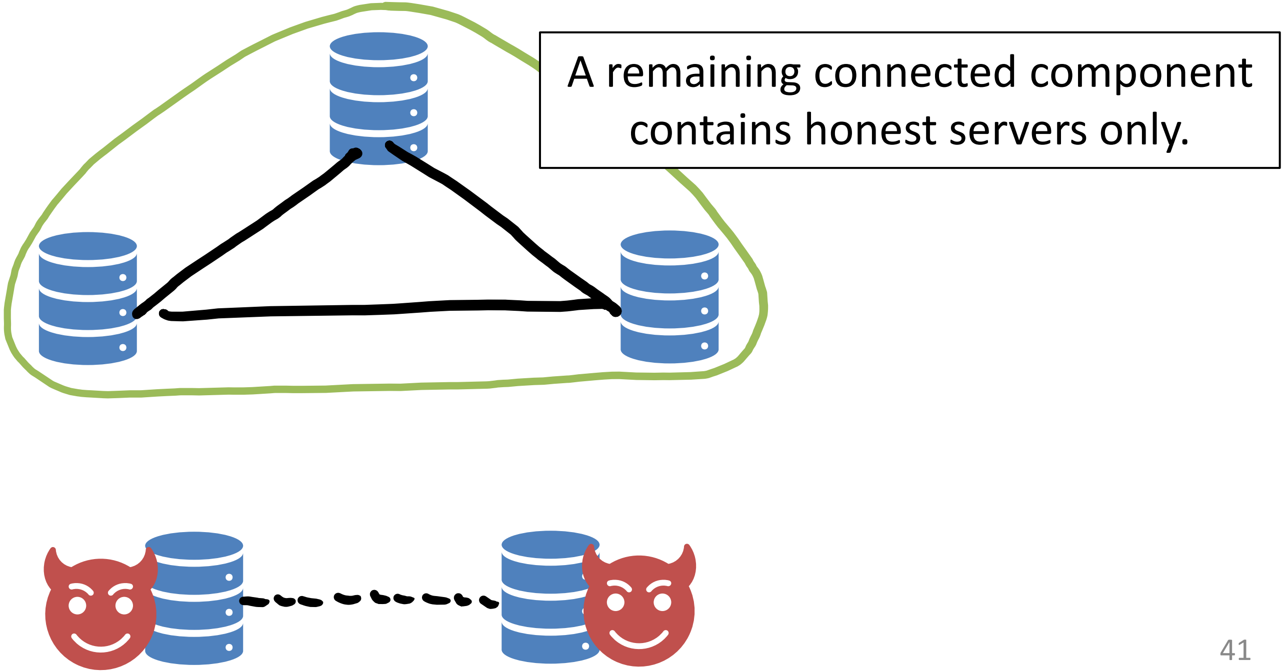
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



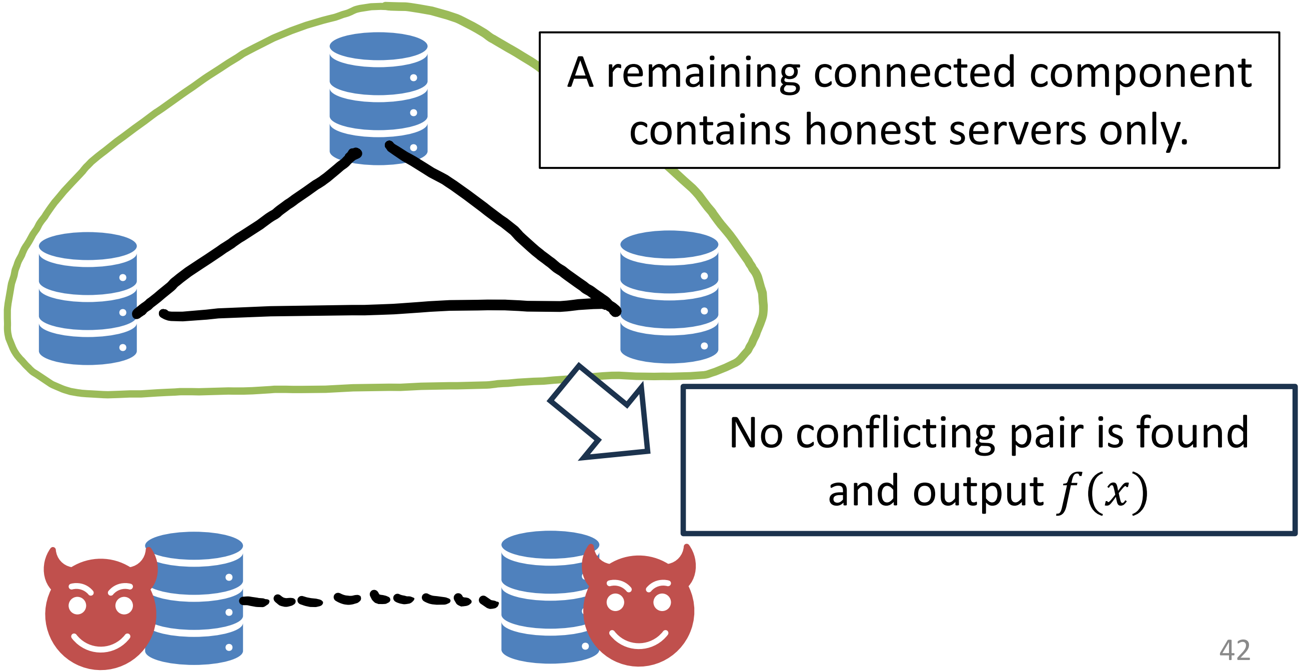
From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



From Conflict-finding to Actively Secure

- The client iterates the following:
  - Choose a connected subgraph of size k and executes a conflict-finding protocol.
  - If a conflicting pair (S_i, S_j) is found, then remove the corresponding edge.



Summary

- Passive-to-active compilers for secure database search protocols with $\text{poly}(m)$ overheads.

Compiler	# servers	# rounds	Function
[BS07]	$m = k + 2t$	1	PIR
[EKN22]	$m = k + t$	1	PIR
Ours-1	$m = k + t$	$O(m^2)$	Any
Ours-2	$m = \Theta(k \log k) + 2t$	1	Any

- Future work
  - Is it possible to achieve $O(1)$ rounds while keeping $m = k + t$?

Reference

[BIW10]: Barkol, O., Ishai, Y., Weinreb, E.: On locally decodable codes, self-correctable codes, and t-private PIR. Algorithmica 58(4).

[BGI16]: Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: Improvements and extensions. ACM CCS ’16.

[BIKO10]: Beimel, A., Ishai, Y., Kushilevitz, E., Orlov, I.: Share conversion and private information retrieval. 2012 IEEE 27th Conference on Computational Complexity.

[WY07]: Woodruff, D., Yekhanin, S.: A geometric approach to information-theoretic private information retrieval. SIAM Journal on Computing 37(4).

[DIJL23]: Dao, Q., Ishai, Y., Jain, A., Lin, H.: Multi-party homomorphic secret sharing and sublinear MPC from sparse LPN. CRYPTO 2023.

[BI05]: Barkol, O., Ishai, Y.: Secure computation of constant-depth circuits with applications to database search problems. CRYPTO 2005.

[CNC+23]: Colombo, S., Nikitin, K., Corrigan-Gibbs, H., Wu, D.J., Ford, B.: Authenticated private information retrieval. USENIX Security 23.

[ZW22]: Zhang, L.F., Wang, H.: Multi-server verifiable computation of low-degree polynomials. 2022 IEEE Symposium on Security and Privacy (SP).

[BS07]: Beimel, A., Stahl, Y.: Robust information-theoretic private information retrieval. Journal of Cryptology 20(3).

[EKN22]: Eriguchi, R., Kurosawa, K., Nuida, K.: On the optimal communication complexity of error-correcting multi-server PIR. In: Theory of Cryptography.

Thank you!