EPFL

# Towards Leakage-Resilient Ratcheted Key Exchange

Daniel Collins, Simone Colombo, **Sina Schaeffler**

Texas A&M University, King's College London, IBM Research/ETH Zurich

PKC, Røros, 13th of May 2025

TEXAS A&M UNIVERSITY

KING'S College LONDON

IBM

ETH zürich

# Plan

1. Introduction
   - Ratcheting
   - Unidirectional ratcheting and its secturity
   - Leakage
2. Security notions for leakage-resilient unidirectional ratcheting
   - Key indistinguishability
   - One-wayness
3. For future

# Ratcheting

## Introduction

RATCHETING: *Key management technique in messaging applications*

Goal:

- AEAD
- Forward security
- Post-compromise security

Idea: Key updates

Example:

- One-way key derivation applied after each send/receive
- DH with fresh ephemeral keys at each complete exchange

# Unidirectional Ratcheted Key Exchange (URKE)

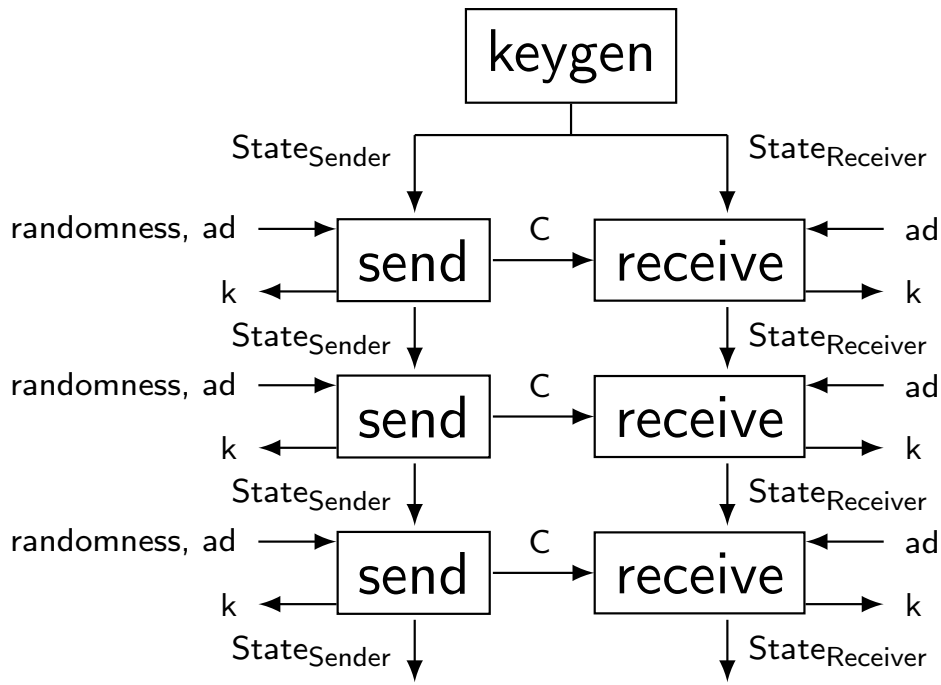
## Introduction

Simplified ratcheting for easier analysis

**Principle:**

Sender *only* sends

Receiver *only* receives




Definition from:
*Determining the Core Primitive for Optimally Secure Ratcheting* by Balli, Rösler, Vaudenay, Asiacrypt 2020

# Balli, Rösler, Vaudenay on URKE without leakage

## Introduction

**Security game for key indistinguishability (KIND) oracles:**

| | | | |
|---|---|---|---|
| Send: | (randomness,ad) | $\rightarrow$ | ctxt |
| Receive: | (ctxt,ad) | $\rightarrow$ | $\perp$ |
| $\text{Expose}_{\text{Sender}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Sender}}$ |
| $\text{Expose}_{\text{Receiver}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Receiver}}$ |
| Reveal: | $\perp$ | $\rightarrow$ | Key |
| Challenge: | id | $\rightarrow$ | real-or-random key |

KIND: Adversary tries to distinguish exchanged keys from random

KIND game and construction from:
*Determining the Core Primitive for Optimally Secure Ratcheting* by Balli, Rösler, Vaudenay, Asiacrypt 2020

# Balli, Rösler, Vaudenay on URKE without leakage

## Introduction

**Security game for key indistinguishability (KIND) oracles:**

| | | | |
|---|---|---|---|
| Send: | (randomness,ad) | $\rightarrow$ | ctxt |
| Receive: | (ctxt,ad) | $\rightarrow$ | $\perp$ |
| $\text{Expose}_{\text{Sender}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Sender}}$ |
| $\text{Expose}_{\text{Receiver}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Receiver}}$ |
| Reveal: | $\perp$ | $\rightarrow$ | Key |
| Challenge: | id | $\rightarrow$ | real-or-random key |

KIND: Adversary tries to distinguish exchanged keys from random

Need to prevent trivial attacks (Example: *$\text{Expose}_{\text{Receiver}}$, Send, Challenge*)

KIND game and construction from:
*Determining the Core Primitive for Optimally Secure Ratcheting* by Balli, Rösler, Vaudenay, Asiacrypt 2020

# Balli, Rösler, Vaudenay on URKE without leakage

## Introduction

**Security game for key indistinguishability (KIND) oracles:**

| | | | |
|---|---|---|---|
| Send: | (randomness,ad) | $\rightarrow$ | ctxt |
| Receive: | (ctxt,ad) | $\rightarrow$ | $\perp$ |
| $\text{Expose}_{\text{Sender}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Sender}}$ |
| $\text{Expose}_{\text{Receiver}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Receiver}}$ |
| Reveal: | $\perp$ | $\rightarrow$ | Key |
| Challenge: | id | $\rightarrow$ | real-or-random key |

KIND: Adversary tries to distinguish exchanged keys from random

Need to prevent trivial attacks (Example: *$\text{Expose}_{\text{Receiver}}$, Send, Challenge*)

**Construction built on**

- SUF-secure MAC (Message Authentication Code)
- KUOW-secure kuKEM (key updatable Key Exchange Mechanism)

KIND game and construction from:
*Determining the Core Primitive for Optimally Secure Ratcheting* by Balli, Rösler, Vaudenay, Asiacrypt 2020

# Leakage Resilience

## Introduction

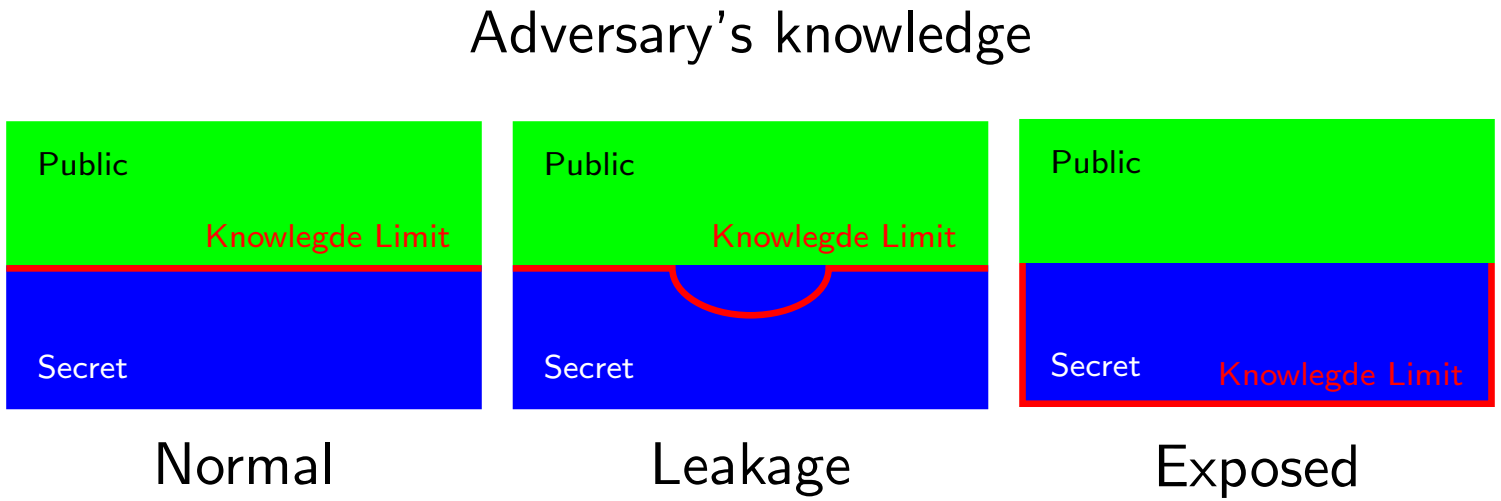
Adversary's knowledge


Normal


Exposed

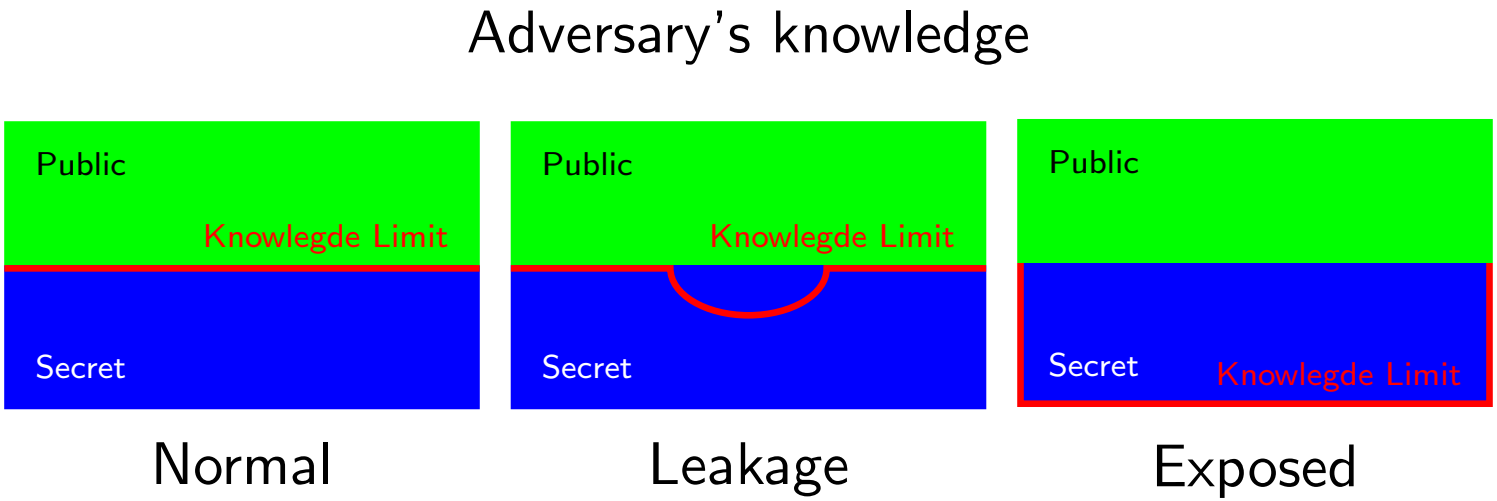
# Leakage Resilience

## Introduction



# Leakage Resilience

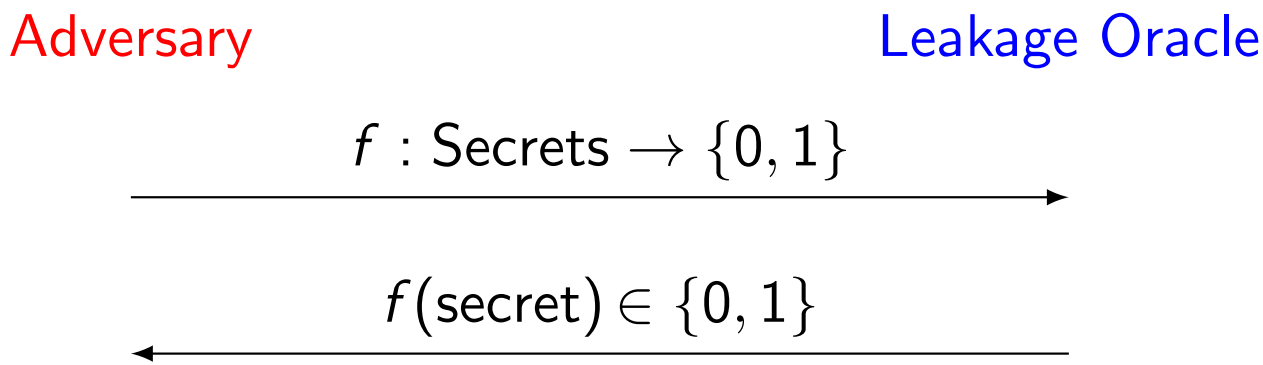
## Introduction




Leakage Resilence (LR): *Theorectical model for side channels*

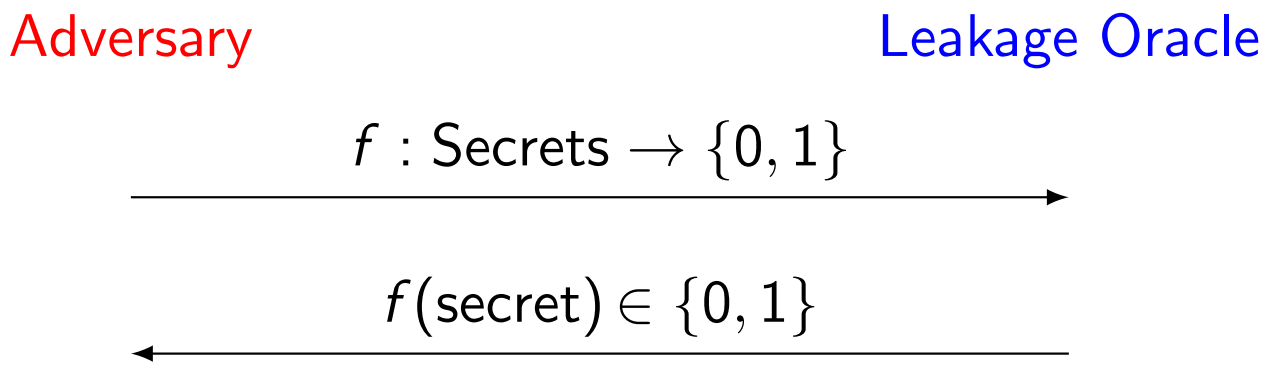
# Leakage Resilience: Choice

## Introduction



# Leakage Resilience: Choice

## Introduction




**When to call it:**

- *Bounded leakage:* Up to $B$ calls to the oracle in entire game
- *Continual leakage:* Up to $B$ calls to the oracle in each interval (for ex. between 2 calls to another oracle)

# LR-KIND definition for URKE

## Security notions for LR-URKE

| | | | |
|---|---|---|---|
| Send: | (randomness,ad) | $\rightarrow$ | ctxt |
| Receive: | (ctxt,ad) | $\rightarrow$ | $\perp$ |
| $\text{Expose}_{\text{Sender}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Sender}}$ |
| $\text{Expose}_{\text{Receiver}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Receiver}}$ |
| Reveal: | $\perp$ | $\rightarrow$ | Key |
| $\text{Leak}_{\text{Sender}}$: | $\text{F}_{\text{State}_{\text{Sender}} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| $\text{Leak}_{\text{Receiver}}$: | $\text{F}_{\text{State}_{\text{Receiver}} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| Challenge: | id | $\rightarrow$ | real-or-random Key |

# LR-KIND definition for URKE

## Security notions for LR-URKE

| | | | |
|---|---|---|---|
| Send: | (randomness,ad) | $\rightarrow$ | ctxt |
| Receive: | (ctxt,ad) | $\rightarrow$ | $\perp$ |
| $\text{Expose}_{\text{Sender}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Sender}}$ |
| $\text{Expose}_{\text{Receiver}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Receiver}}$ |
| Reveal: | $\perp$ | $\rightarrow$ | Key |
| $\text{Leak}_{\text{Sender}}$: | $F_{\text{State}_{\text{Sender}} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| $\text{Leak}_{\text{Receiver}}$: | $F_{\text{State}_{\text{Receiver}} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| Challenge: | id | $\rightarrow$ | real-or-random Key |

- Receive is deterministic and only depends on :
  - The initial $\text{State}_{\text{Receiver}}$
  - All received ciphertexts (public)

# LR-KIND definition for URKE

## Security notions for LR-URKE

| | | | |
|---|---|---|---|
| Send: | (randomness,ad) | $\rightarrow$ | ctxt |
| Receive: | (ctxt,ad) | $\rightarrow$ | $\perp$ |
| $\text{Expose}_{\text{Sender}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Sender}}$ |
| $\text{Expose}_{\text{Receiver}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Receiver}}$ |
| Reveal: | $\perp$ | $\rightarrow$ | Key |
| $\text{Leak}_{\text{Sender}}$: | $F_{\text{State}_{\text{Sender}} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| $\text{Leak}_{\text{Receiver}}$: | $F_{\text{State}_{\text{Receiver}} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| Challenge: | id | $\rightarrow$ | real-or-random Key |

- Receive is deterministic and only depends on :
  - The initial $\text{State}_{\text{Receiver}}$
  - All received ciphertexts (public)

**Trivial attack:**

1. Call $\text{Leak}_{\text{Receiver}}$ with (iterated) decrypt function (using output of Send oracle)
2. Learn a bit of the key $\rightarrow$ (likely) win KIND

$\longrightarrow$ **Need to forbid all leakage on Receiver!**

# Advantages and issues of LR-KIND security

## Security notions for LR-URKE

+ Can leak in one case we cannot expose
+ Continual leakage

# Advantages and issues of LR-KIND security

## Security notions for LR-URKE

+ Can leak in one case we cannot expose
+ Continual leakage
+ Simple construction
  - *Bounded* LR-SUF MAC
  - *Non-LR* kuKEM

# Advantages and issues of LR-KIND security

## Security notions for LR-URKE

+ Can leak in one case we cannot expose
+ Continual leakage
+ Simple construction
  - *Bounded* LR-SUF MAC
  - *Non-LR* kuKEM
− No leakage on receiver is not realistic
− KIND and leakage seem hard to achieve simultaneously

→ What about a one-wayness security definition for LR-URKE?

# LR-OW definition for URKE

## Security notions for LR-URKE

| | | | |
|---|---|---|---|
| Send: | (randomness,ad) | $\rightarrow$ | ctxt |
| Receive: | (ctxt,ad) | $\rightarrow$ | $\perp$ |
| $\text{Expose}_{\text{Sender}}$: | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Sender}}$ |
| $\text{Expose}_{\text{Receiver}}$ | $\perp$ | $\rightarrow$ | $\text{State}_{\text{Receiver}}$ |
| $\text{Reveal}_{\text{Key}}$: | $\perp$ | $\rightarrow$ | Key |
| $\text{Leak}_{\text{Sender}}$: | $F_{\text{State}_{\text{Sender}} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| $\text{Leak}_{\text{Receiver}}$: | $F_{\text{State}_{\text{Receiver}} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| $\text{Leak}_{\text{Key}}$: | $F_{\text{Key} \rightarrow \{0,1\}}$ | $\rightarrow$ | $\{0,1\}$ |
| Challenge: | (guessed key, id) | $\rightarrow$ | $\perp$ |

OW: Adversary tries to guess an exchanged key

$\longrightarrow$ Trivial attacks include leaking too much on Receiver

# LR-OW URKE constructions

## Security notions for LR-URKE

**Construction**

- *Bounded* LR-SUF MAC
- *Bounded* LR-KUOW kuKEM
- The obtained scheme is also LR-KIND!

**LR-KUOW**

- LR-KUOW: KOUW game with bounded leakage on secret key
- Reduction to LR-HIBE (Leakage-Resilient Hierarchical Identity-Based Encryption)
- No suitable LR-HIBE construction exists

# Conclusions on LR-OW

## Security notions for LR-URKE

+ Allows for more leakage than KIND
+ Allows some leakage on exchanged keys

# Conclusions on LR-OW

## Security notions for LR-URKE

+ Allows for more leakage than KIND
+ Allows some leakage on exchanged keys

– Still bounded leakage on Receiver
– No KIND security any more
– No full construction

# Future directions

## Conclusion

- Construct the missing LR-HIBE
- Investigate other leakage models
- Extend to bidirectional ratcheting
  - Challenge: everyone receives
  - Advantage: everyone sends

eprint 2025/332