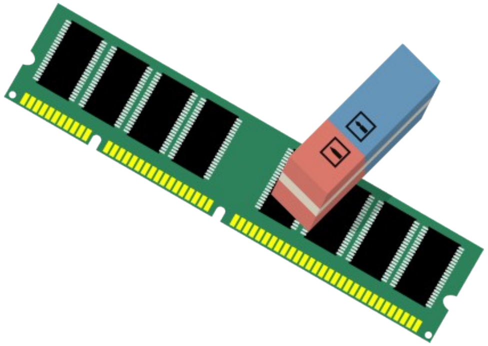
The 10th Conference on Security and Cryptography for Networks (SCN) – August 31, 2016

Memory Erasability Amplification



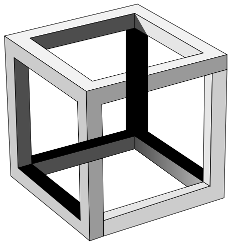
Jan Camenisch
IBM Research–Zurich

Robert R. Enderlein – **scn2016@e7n.ch**
work carried out while at IBM Research–Zurich and ETH Zurich

Ueli Maurer
ETH Zurich

Goal: Practical Protocols with Strong Security

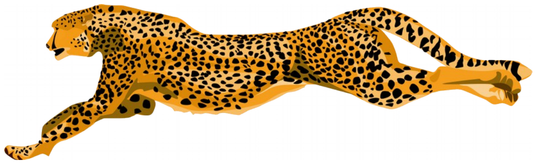
- Realistic assumptions.
  No random oracles. Allow CRS.



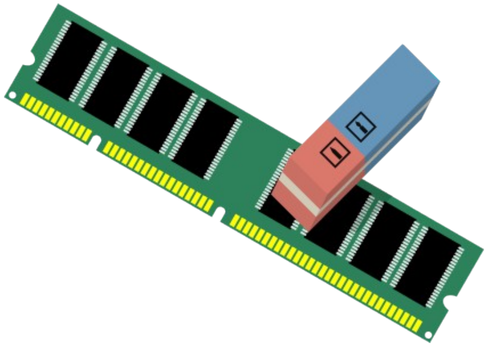
- Provably secure in arbitrary contexts.
  Designed in a composition framework
  (UC, GNUC, Abstract Cryptography, ...).

- Secure against adaptive adversaries.
  Real computers can be compromised at any time.

- Efficient enough for practical settings.



The Need for Erasable Memory



- Erasable memory crucial for most practical adaptively secure protocols.
- Not always available in reality:
  - –Computers designed to preserve data, not erase it.
  - –File systems don't erase deleted files; keep traces in journal.
  - –SSD's don't flash blocks containing overwritten data right away.
- Important to model imperfectly erasable memory.
  - –Attempt by [CEGL08, Lim08], but needed to change framework.
- Re-use existing protocols by constructing perfect memory from imperfect one.

[CEGL08]: Canetti, Eiger, Goldwasser, Lim.
How to Protect Yourself without Perfect Shredding. *ICALP 2008.*
[Lim08]: Lim. *The Paradigm of Partial Erasures*. PhD thesis, MIT, 2008.

Our Contributions

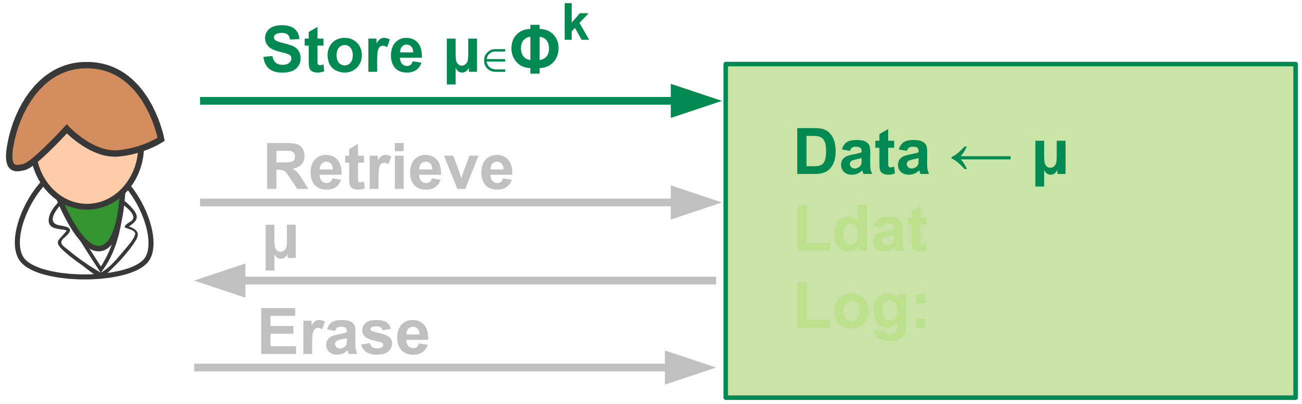
- We formally model imperfectly erasable memory in the Abstract Cryptography (AC) framework.

- We investigate how to amplify the erasability of such memories.

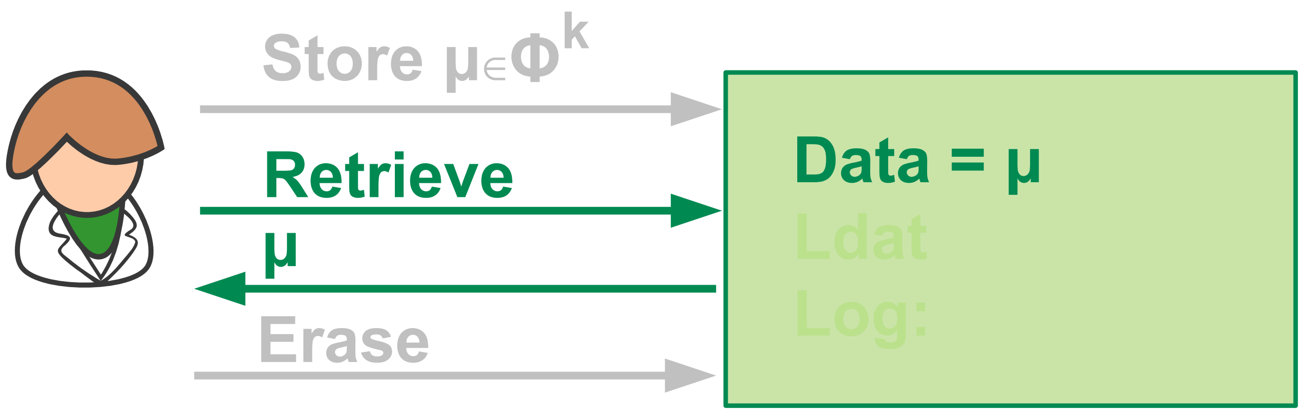
- We propose better constructions of All-or-Nothing Transforms (AoNTs).
  (Not in today's slides: see the paper.)

Modeling Erasable Memory in the AC Framework

- Memory can be written once.
  - If multiple writes: use multiple resources.

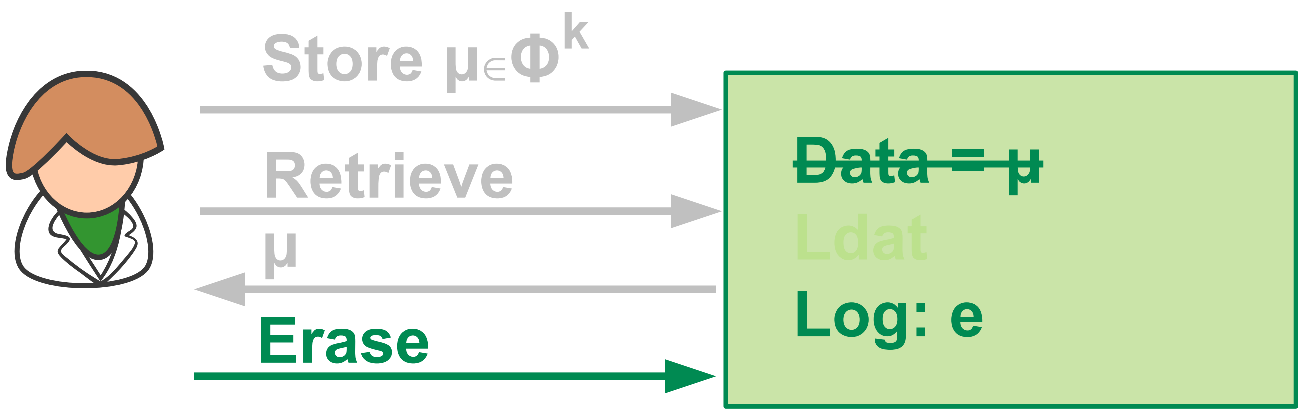


Modeling Erasable Memory in the AC Framework



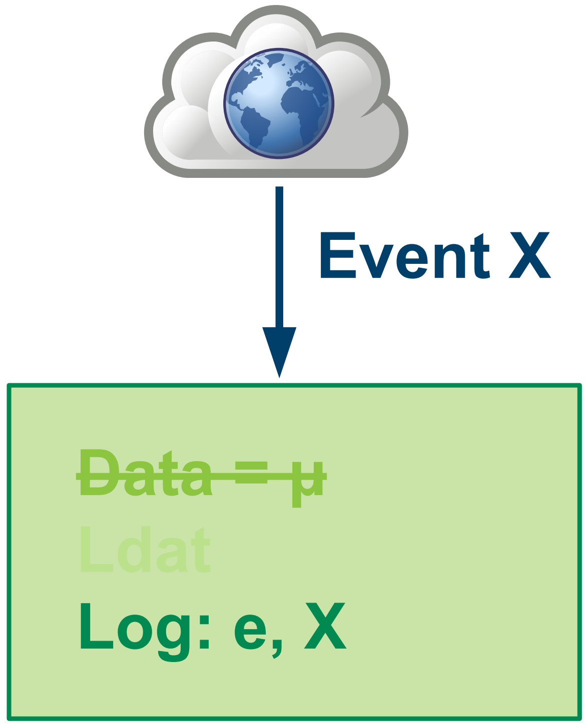
Modeling Erasable Memory in the AC Framework

- Entire memory is erased.
  – For more granularity: use multiple resources.



- Erasure event is logged.

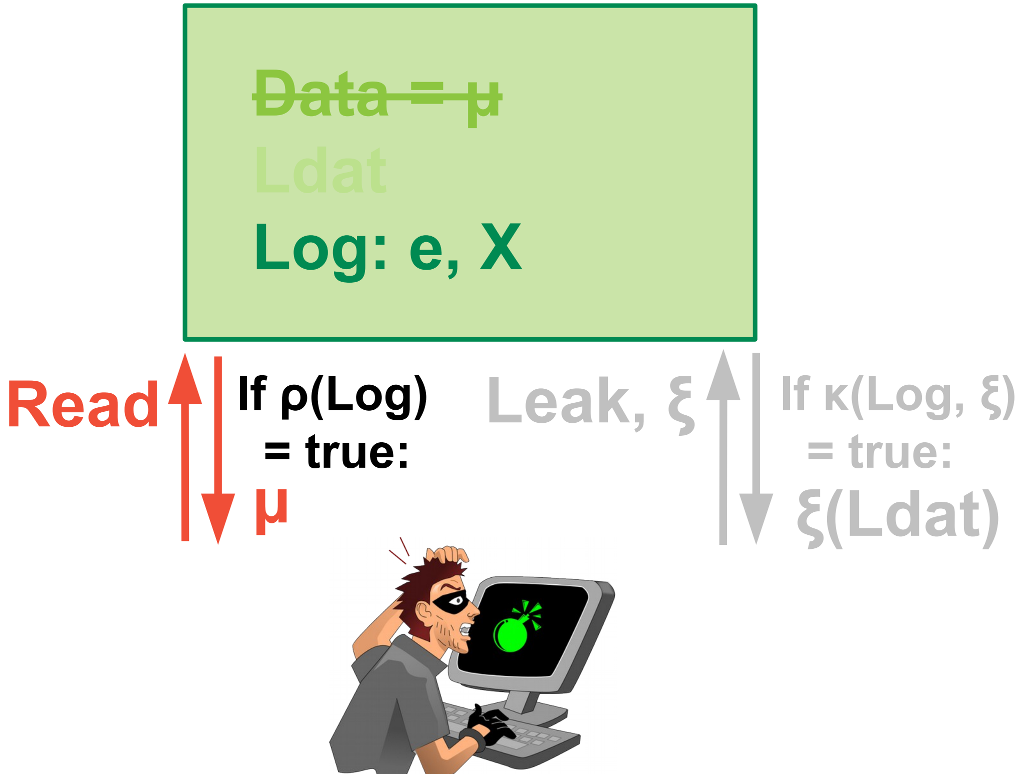
Modeling Erasable Memory in the AC Framework



- Environment can influence resource through events.
  - Malware, adversary gets physical access, or even environmental conditions.
  - Events not triggered by the adversary: otherwise no checks & balances.
- Security guarantees of resource depends on those events.
- Events are logged.

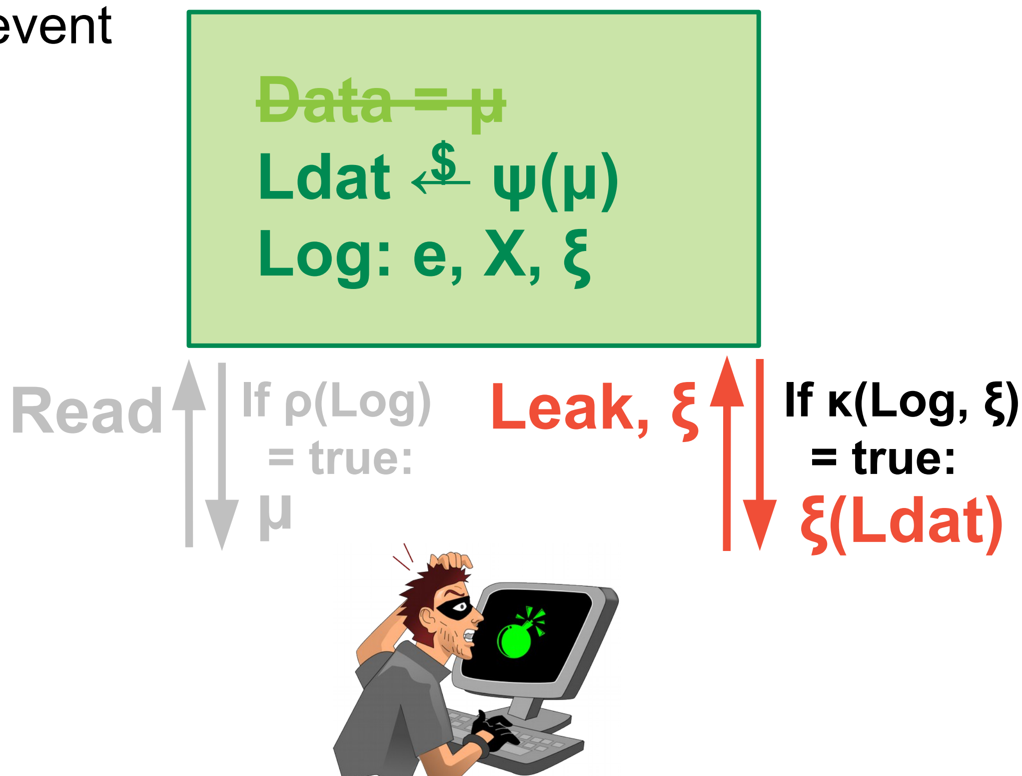
Modeling Erasable Memory in the AC Framework

- Adversarial access: none, total (Read), or partial (Leak).
- Total access if predicate **ρ** on event log is true.
  - Typically: “critical” event before/without erasure.



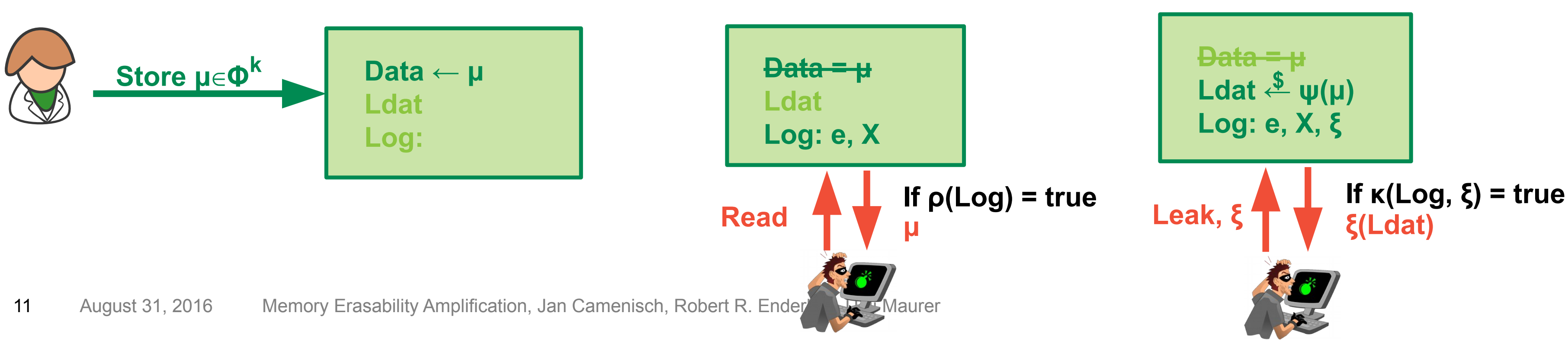
Modeling Erasable Memory in the AC Framework

- Adversary might influence result: deterministic function ξ.
- Potential leakage **Ldat** dependent on random function **ψ**.
- Gets **ξ(Ldat)=ξ(ψ(μ))** if predicate **κ** on event log & **ξ** is true.
  - Typically: “critical” event after erasure and **ξ** is OK.
- Adaptive queries.



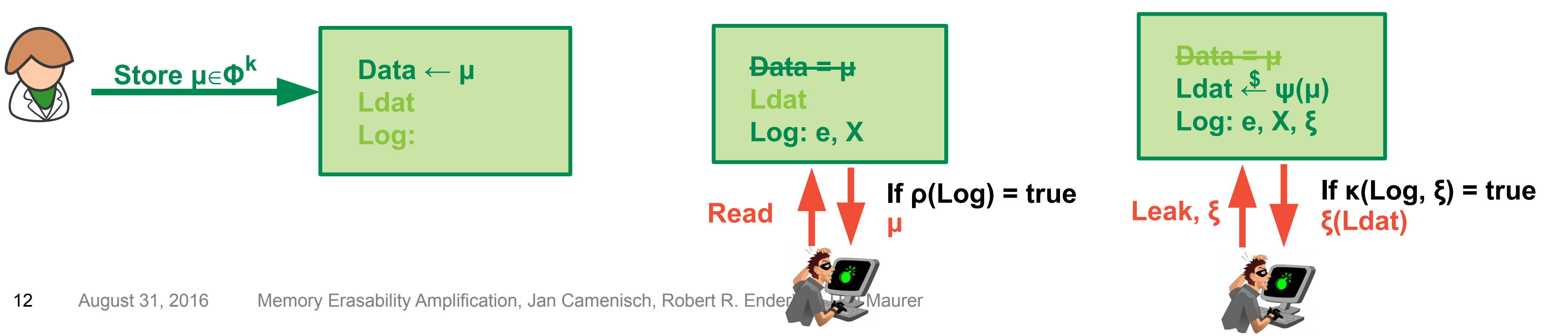
Types of Erasable Memory

- Typical types of memory are just specializations:
  - –Perfectly erasable memory.
    - **ρ** is true if memory was attacked before/without erase.
    - **κ** returns false.
  - –Imperfectly erasable memory:
    - ♦ Memory leaking a constant number of bits.
      - **ρ** idem.
      - **ψ**(**μ**)=**μ**.
      - **κ** is true if **Log**=(**e**, **X**) and **ξ** reads **d** bits of **Ldat** (and thus of **μ**).
    - ♦ Memory leaking a noisy version of the data.
  - –Non-erasable memory.



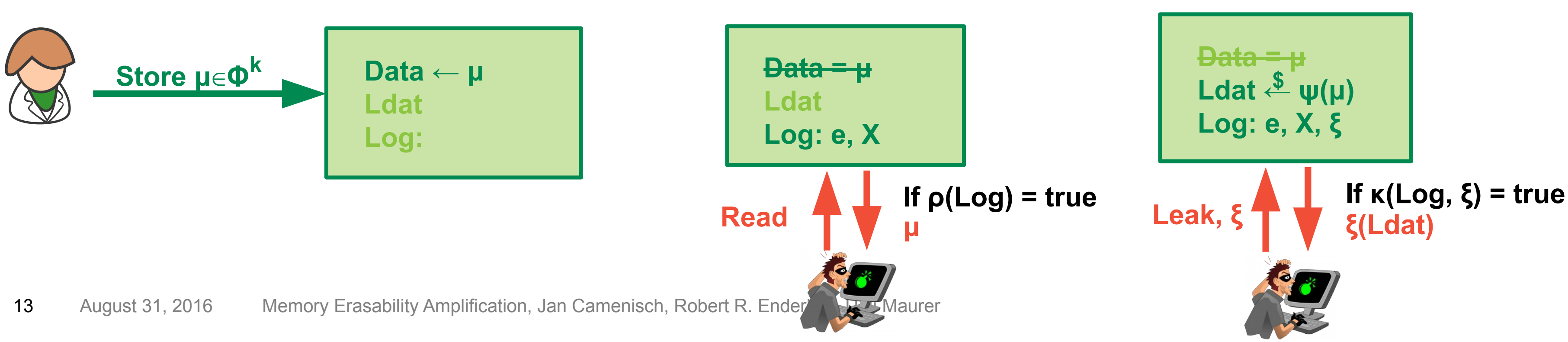
Types of Erasable Memory

- Typical types of memory are just specializations:
  - –Perfectly erasable memory.
    - **ρ** is true if memory was attacked before/without erase.
    - **κ** returns false.
  - –Imperfectly erasable memory:
    - ♦ Memory leaking a constant number of bits.
      - **ρ** idem.
      - **ψ**(**μ**)=**μ**.
      - **κ** is true if **Log**=(**e**, **X**) and **ξ** reads **d** bits of **Ldat** (and thus of **μ**).
    - ♦ Memory leaking a noisy version of the data.
  - –Non-erasable memory.



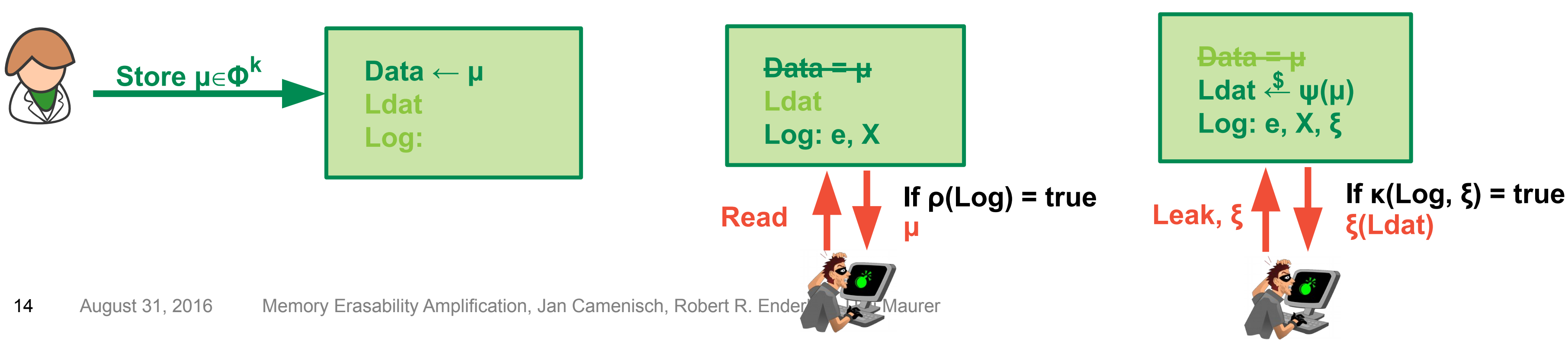
Types of Erasable Memory

- Typical types of memory are just specializations:
  - –Perfectly erasable memory.
    - **ρ** is true if memory was attacked before/without erase.
    - **κ** returns false.
  - –Imperfectly erasable memory:
    - ♦ Memory leaking a constant number of bits.
      - **ρ** idem.
      - **ψ(μ)=μ**.
      - **κ** is true if **Log**=(**e**, **X**) and **ξ** reads **d** bits of **Ldat** (and thus of **μ**).
    - ♦ Memory leaking a noisy version of the data.
  - –Non-erasable memory.



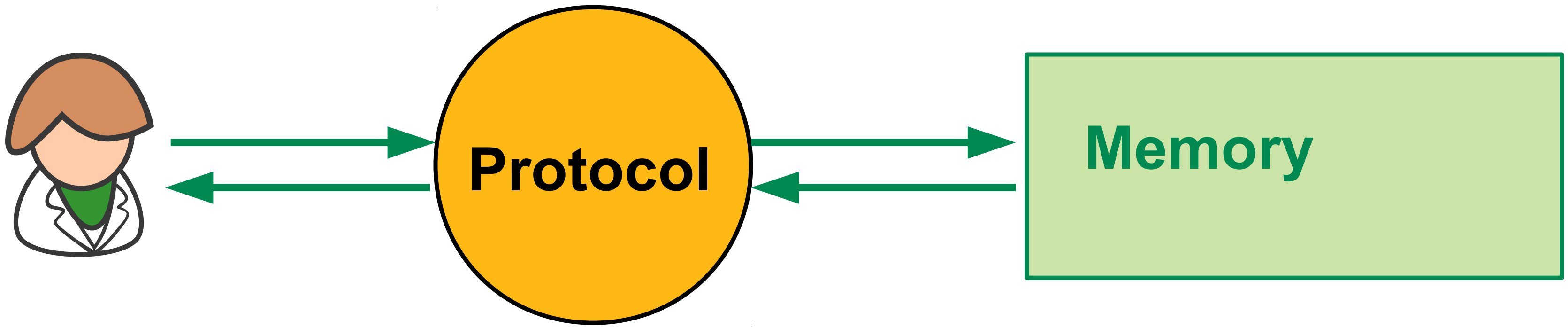
Types of Erasable Memory

- Typical types of memory are just specializations:
  - Perfectly erasable memory.
    - **ρ** is true if memory was attacked before/without erase.
    - **κ** returns false.
  - Imperfectly erasable memory:
    - ♦ Memory leaking a constant number of bits.
      - **ρ** idem.
      - **ψ(μ)=μ**.
      - **κ** is true if **Log**=(**e**, **X**) and **ξ** reads **d** bits of **Ldat** (and thus of **μ**).
    - ♦ Memory leaking a noisy version of the data.
  - Non-erasable memory.

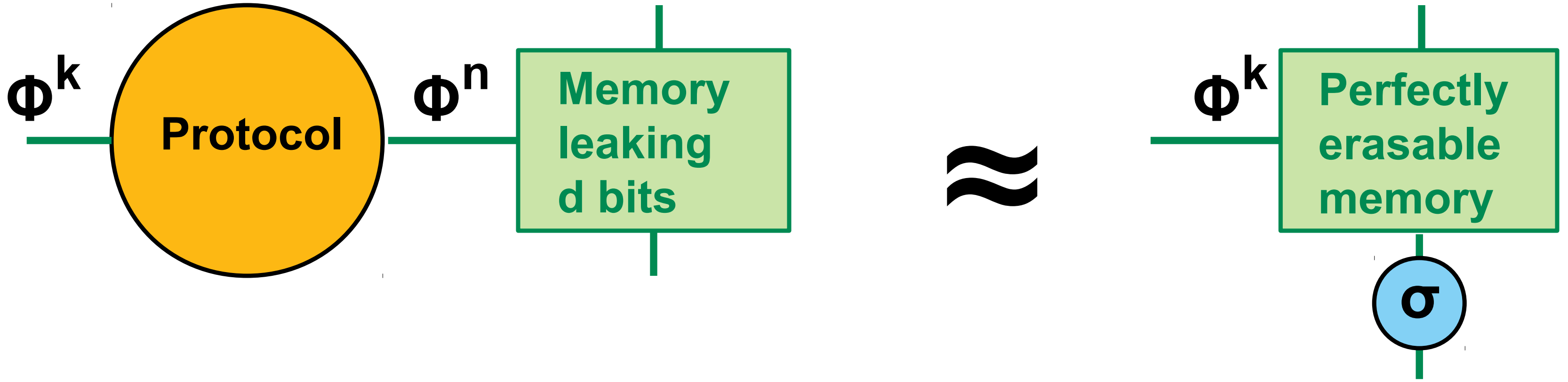


Building Protocols using our Memory

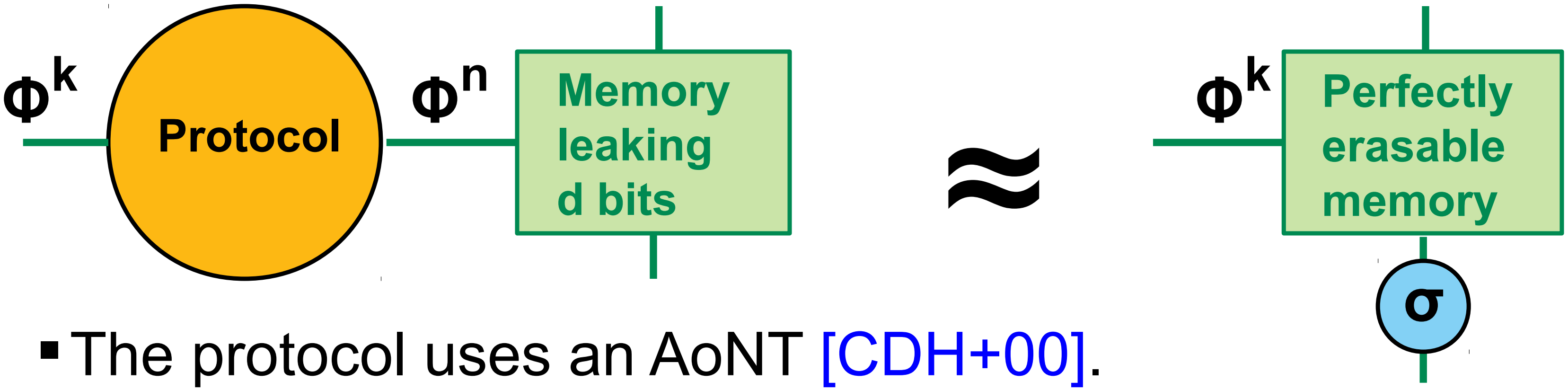
- Goal: protocols that work with imperfectly erasable memory.
- Protocols must not circumvent the memory resource:
  - Maintain no internal state between computation phases.
  - But can use temporary storage (registers) during phase (to avoid strong dependency on actual implementation).



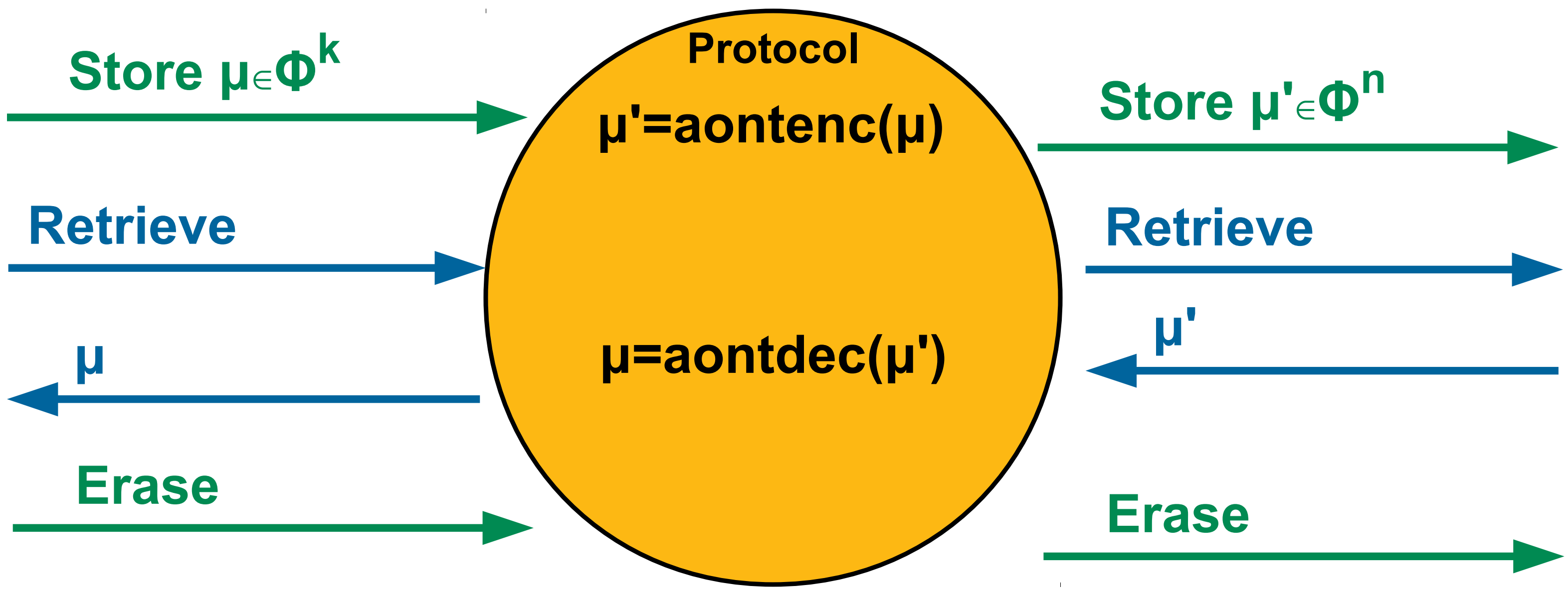
Constructing Perfectly Erasable Memory



Constructing Perfectly Erasable Memory



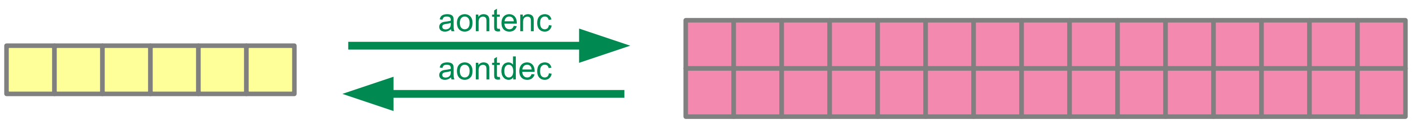
- The protocol uses an AoNT [CDH+00].



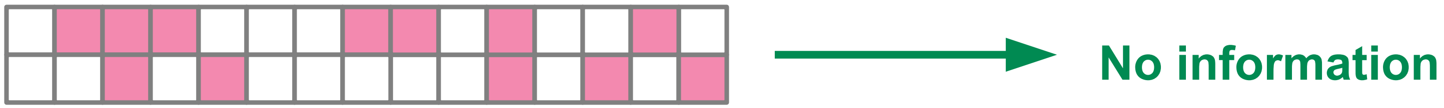
[CDH+00]: Canetti, Dodis, Halevi, Kushilevitz, Sahai. Exposure-Resilient Functions and All-or-Nothing Transforms. *Eurocrypt 2000*.

All-or-Nothing Transform [CDH+00]

- Completeness:
  - $\forall \boldsymbol{\mu} \in \boldsymbol{\Phi}^{\mathbf{k}}$: $\boldsymbol{\mu}$ = aontdec(aontenc($\boldsymbol{\mu}$)).



- Privacy:
  - For all sets **L** of size **d**, $\boldsymbol{\mu}_{\mathbf{0}} \in \boldsymbol{\Phi}^{\mathbf{k}}$, $\boldsymbol{\mu}_{\mathbf{1}} \in \boldsymbol{\Phi}^{\mathbf{k}}$:
    $(\boldsymbol{\mu}_{\mathbf{0}}, \boldsymbol{\mu}_{\mathbf{1}}, [\text{aontenc}(\boldsymbol{\mu}_{\mathbf{0}})]_{\mathbf{L}}) \approx (\boldsymbol{\mu}_{\mathbf{0}}, \boldsymbol{\mu}_{\mathbf{1}}, [\text{aontenc}(\boldsymbol{\mu}_{\mathbf{1}})]_{\mathbf{L}})$.

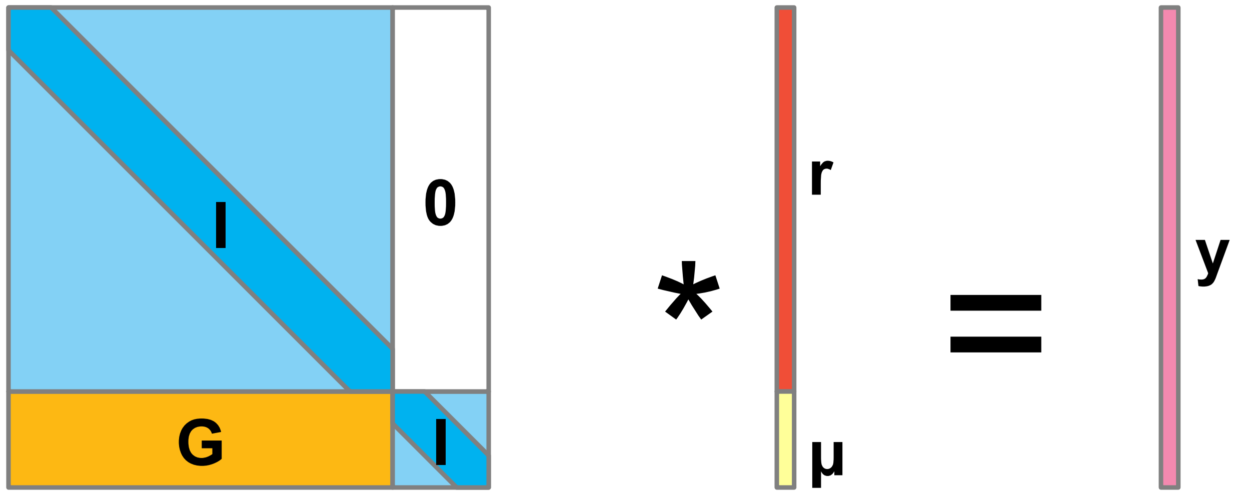


[CDH+00]: Canetti, Dodis, Halevi, Kushilevitz, Sahai. Exposure-Resilient Functions and All-or-Nothing Transforms. *Eurocrypt 2000*.

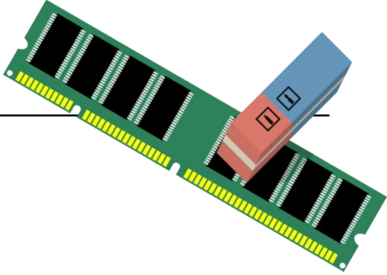
Examples of AoNT

- (Ramp) secret sharing scheme:
  - –Based on Shamir secret sharing (only for large **Φ**). [BM84]
  - –For **Φ**={0, 1}, construction using linear block code. [CDH+00]

Generator matrix **G** of minimum distance **d**.



[BM84]: Blakley, Meadows. Security of Ramp Schemes. *Crypto 1984*.



Summary

- Erasable memory crucial for most practical adaptively secure protocols.
- Not always available in reality
  → Important to model imperfect memory.

- We provided a formal model of erasable memory in the Abstract Cryptography (AC) framework.
- We Investigated how to amplify the erasability of such memories.
- We proposed better All-or-Nothing Transforms (AoNTs).

Thank you!

Contact e-mail: **scn2016@e7n.ch**