Advanced Cryptography
CS 655

Week 3:

- Memory-Tight Reductions
- RSA-FDH
- Memory-Tightness

Memory-Tight Reductions

Benedikt Auerbach[1] David Cash[2] Manuel Fersch[1] Eike Kiltz[1]

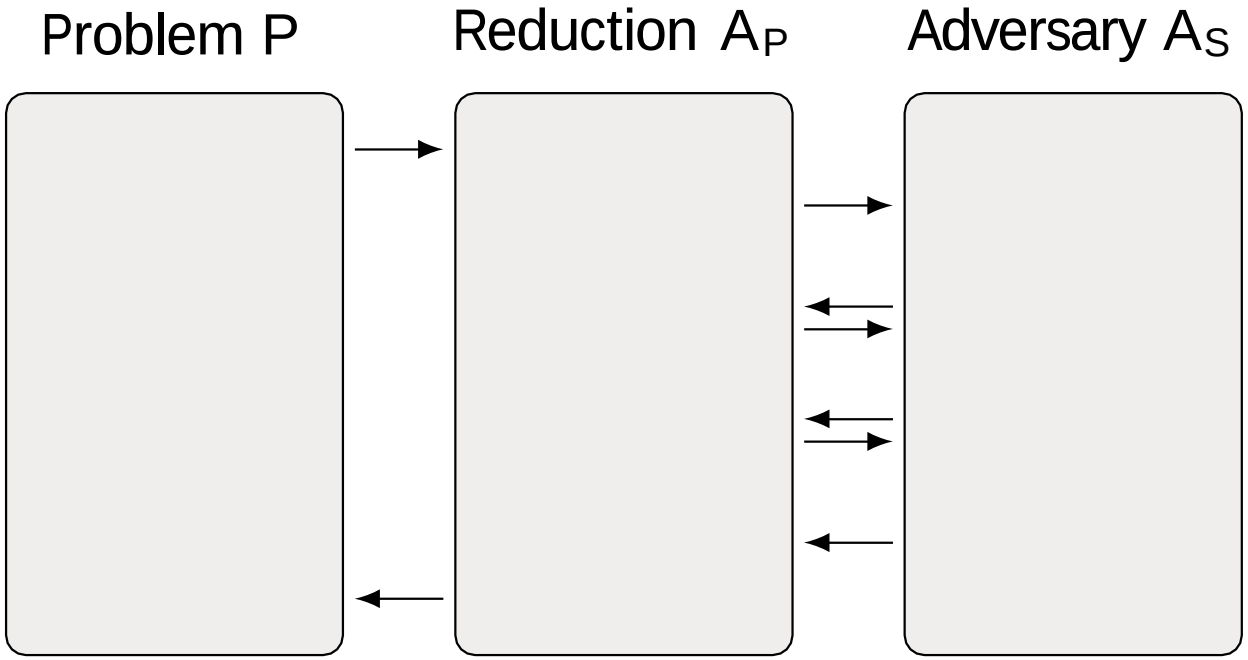
[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany

[2] Rutgers University, New Jersey, USA

Crypto 2017, Santa Barbara
August 21, 2017

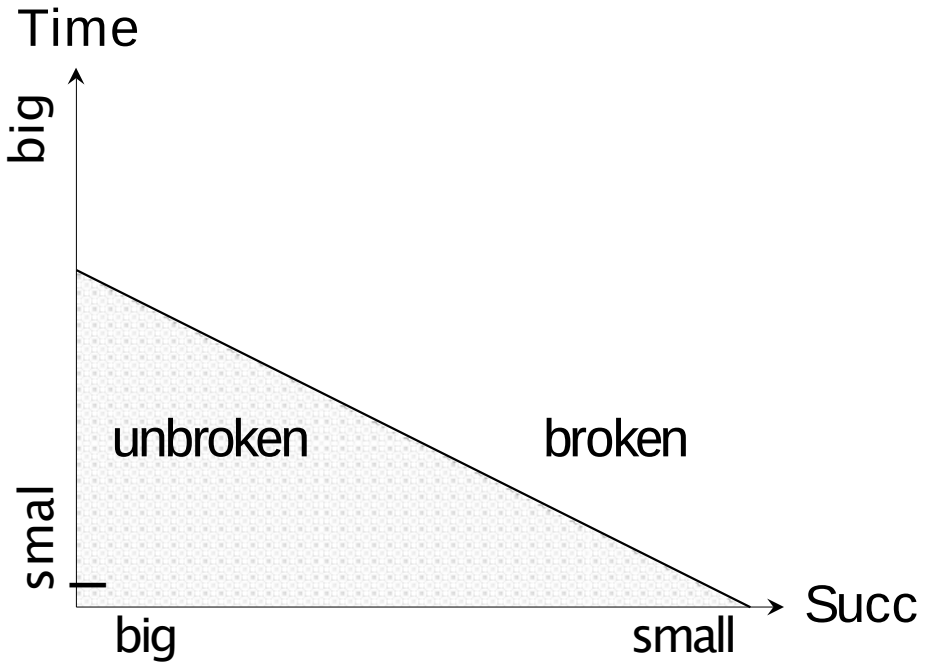
Tightness and Memory-Tightness

Cryptographic Reductions and Tightness



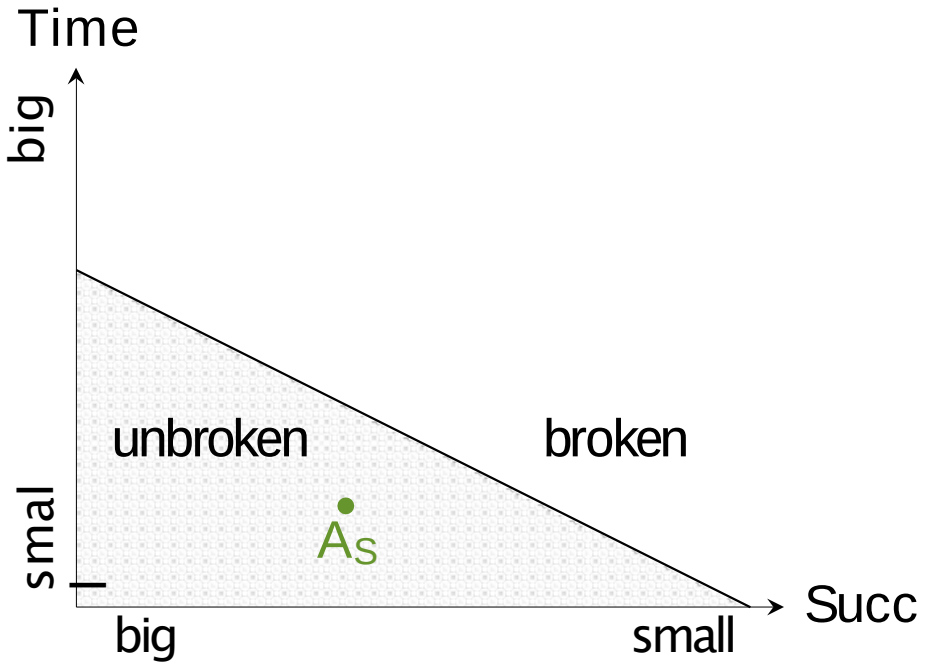
Reduction is *tight* if $\mathrm{Time}(A_P) \approx \mathrm{Time}(A_S)$ and $\mathrm{Succ}(A_P) \approx \mathrm{Succ}(A_S)$

Relevance of Tightness for Concrete Security



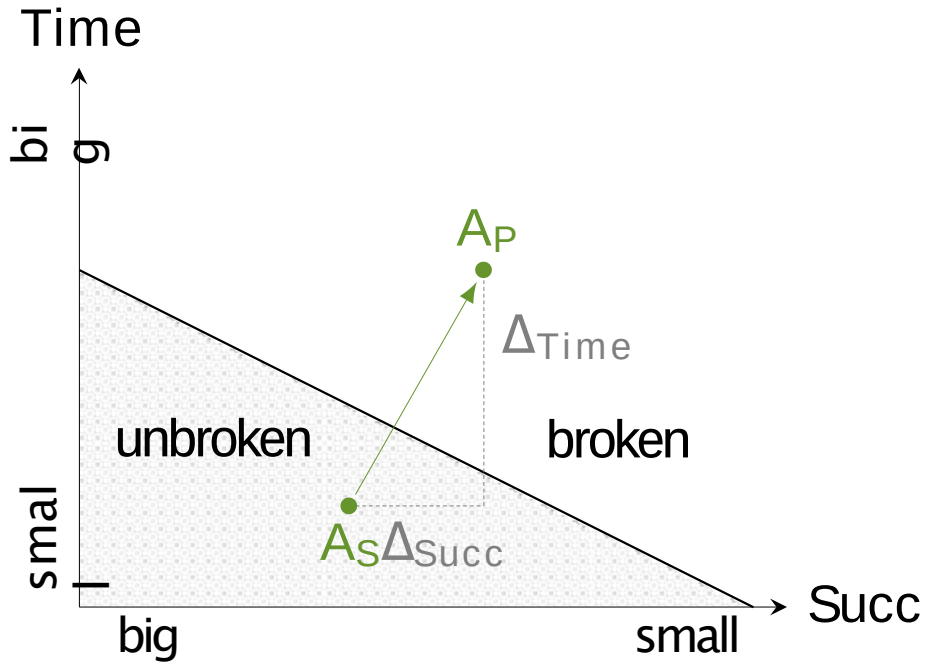
Time-success trade-off plot for algorithms solving problem P

Relevance of Tightness for Concrete Security



Time-success trade-off plot for algorithms solving problem P

Relevance of Tightness for Concrete Security



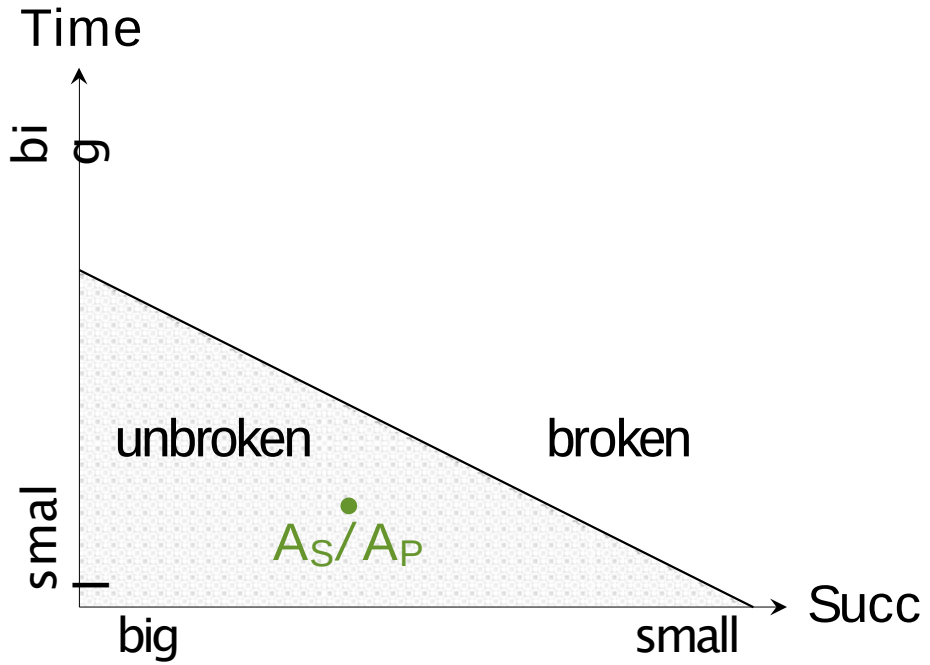
Non-tight reduction

$$\Delta_{Succ} = Succ(A_S)/Succ(A_P) \gg 1$$

$$\text{or } \Delta_{Time} = Time(A_P)/Time(A_S) \gg 1$$

Benedikt Auerbach: Memory-Tight Reductions

Relevance of Tightness for Concrete Security



Tight reduction

$$\mathsf{Succ}(A_P) \approx \mathsf{Succ}(A_S)$$
$$\mathsf{Time}(A_P) \approx \mathsf{Time}(A_S)$$

Benedikt Auerbach: Memory-Tight Reductions

What about memory?

Resources of adversary

- Running time
- Success probability
- Memory consumption

Contributions

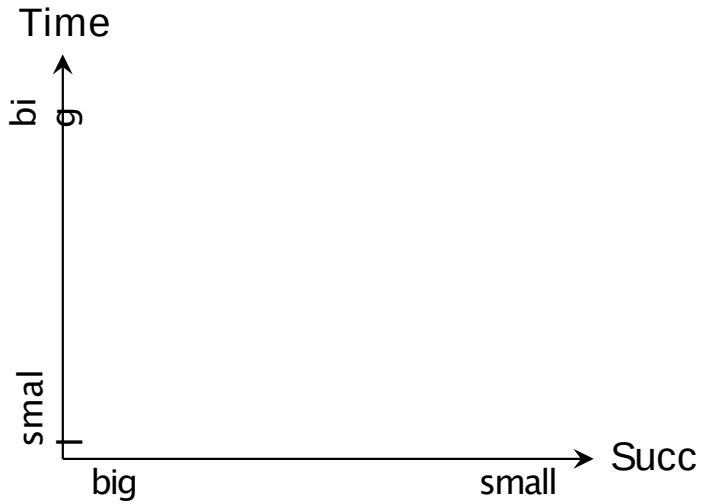
- Raise awareness for memory usage in reductions
- Propose tools for achieving memory-tightness
- Concrete application: memory-tight reduction for RSA-FDH
- Impossibility of memory-tight reductions for certain problems

Time-Memory Trade-offs

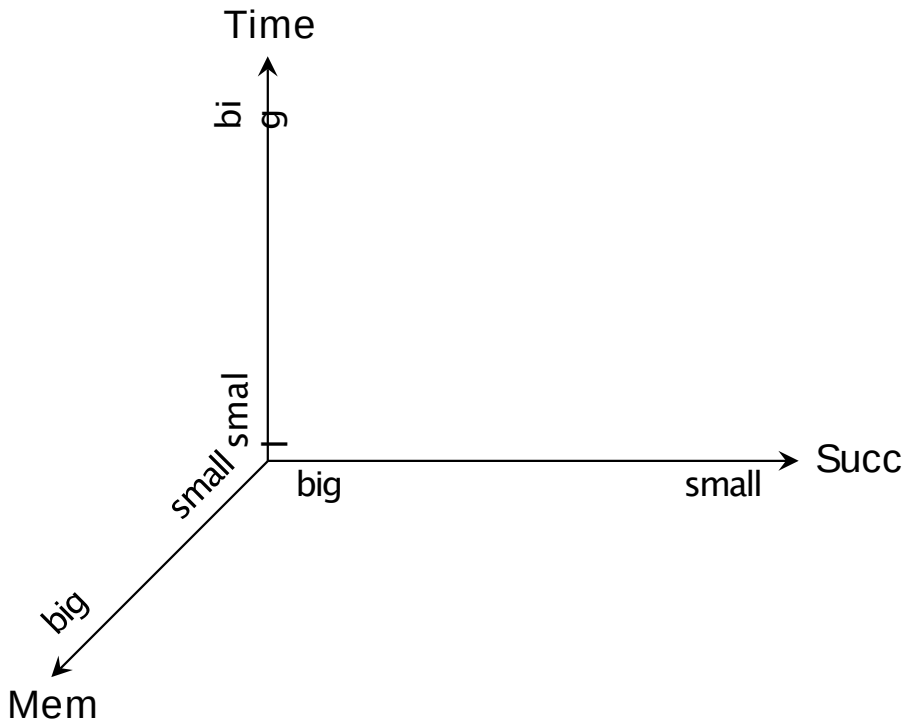
Time-Memory Trade-offs

- Some problems are harder with less memory
  - in particular many lattice / coding problems
- Needs to be taken into account in reductions
- Concrete example: Learning Parity with Noise (LPN)

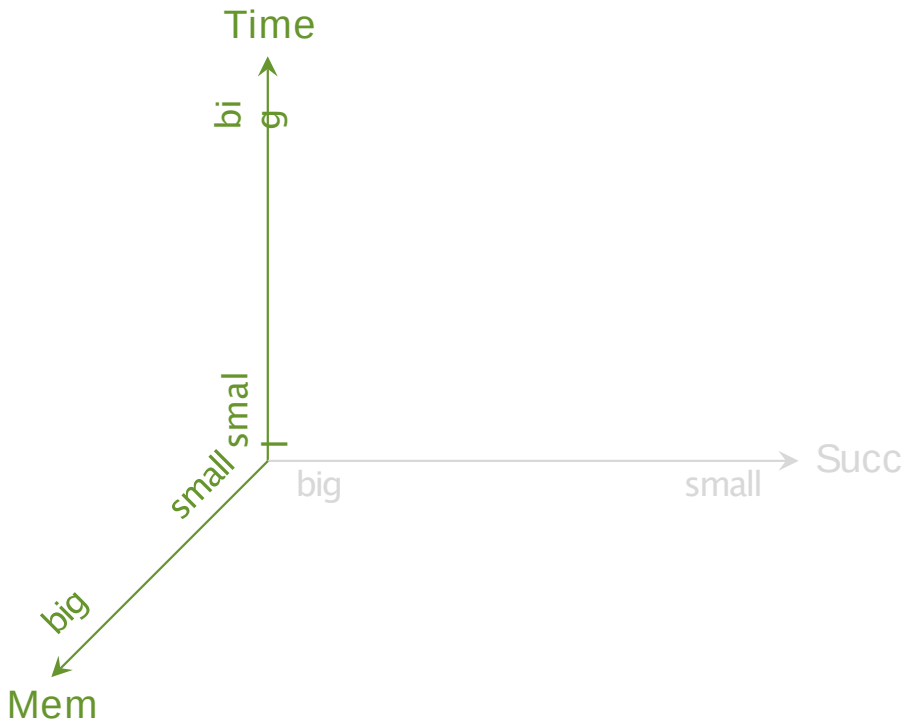
Adding Memory-Consumption



Adding Memory-Consumption

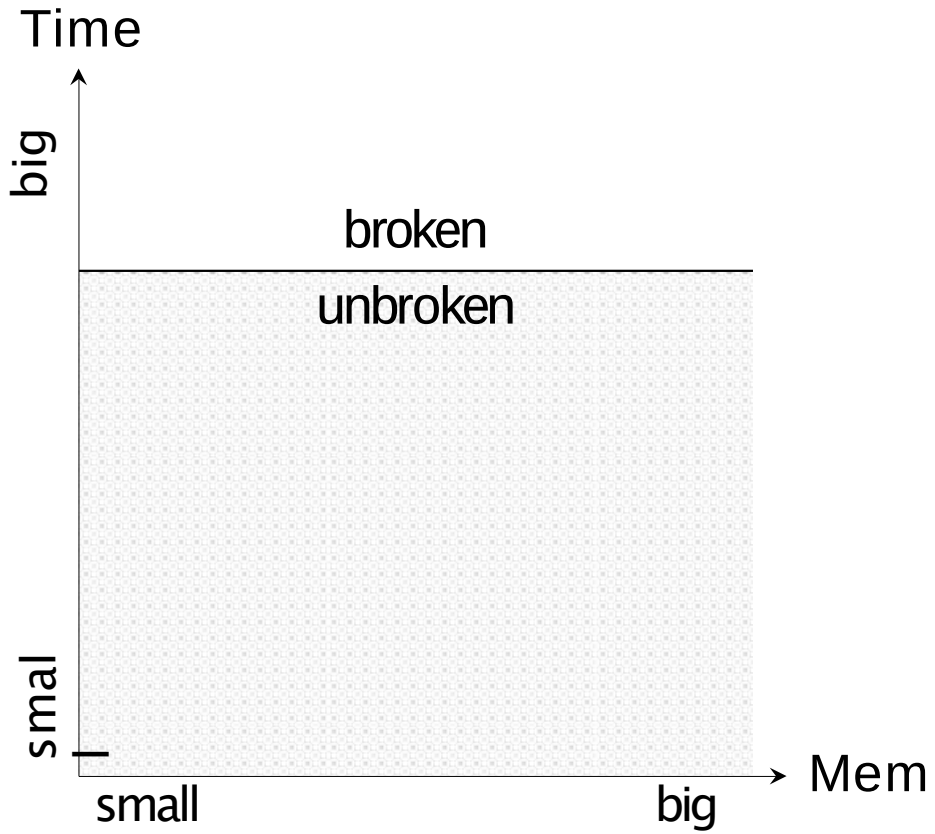


Adding Memory-Consumption



For simplicity: consider algorithms with constant success probability

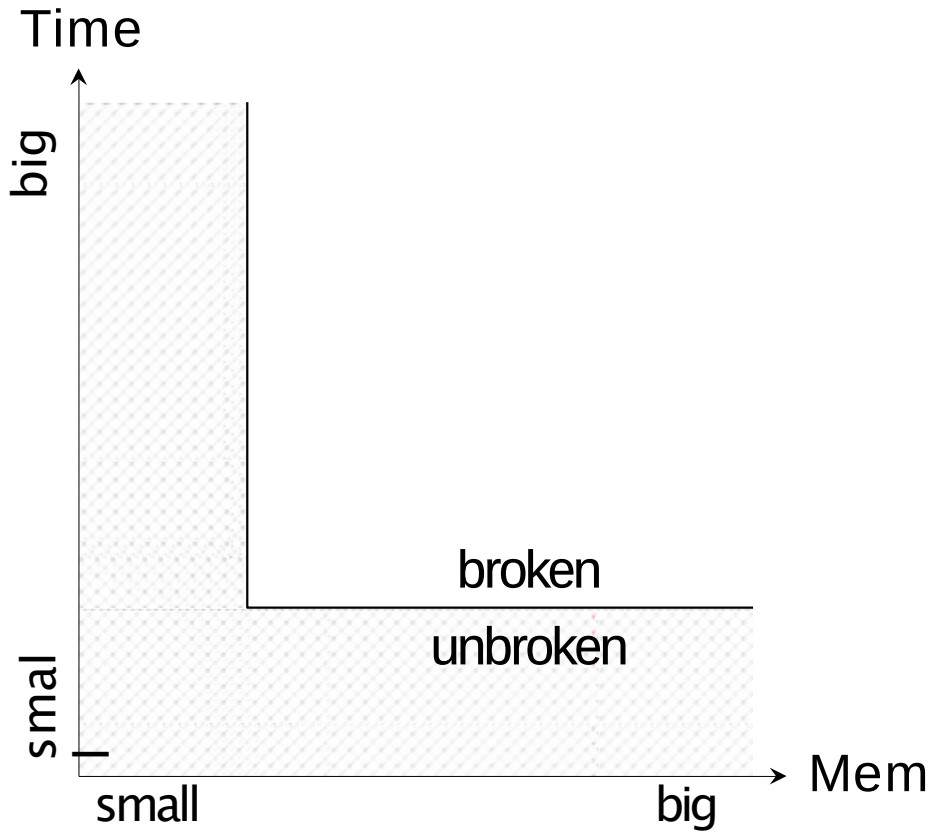
Example: Time-Memory Trade-off for LPN



Algorithm 1: Gauss algorithm for dimension λ

$\mathrm{Mem} = \mathrm{poly}(\lambda),\ \mathrm{Time} \approx 2^{\lambda}$

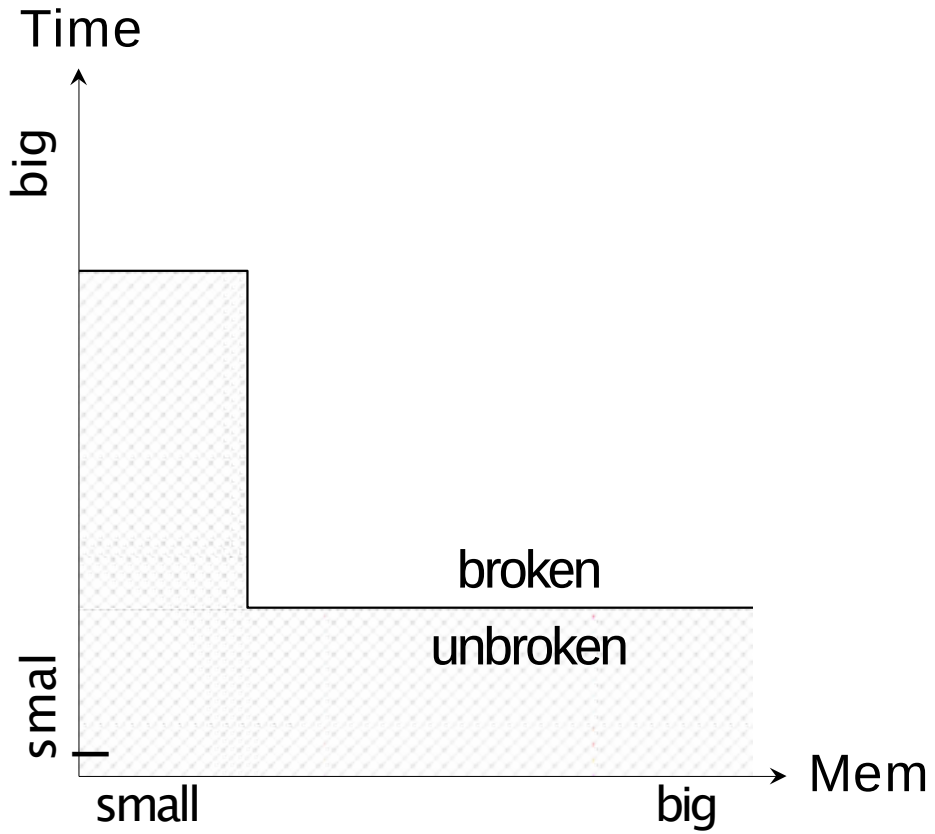
Example: Time-Memory Trade-off for LPN



Algorithm 2: BKW algorithm for dimension λ

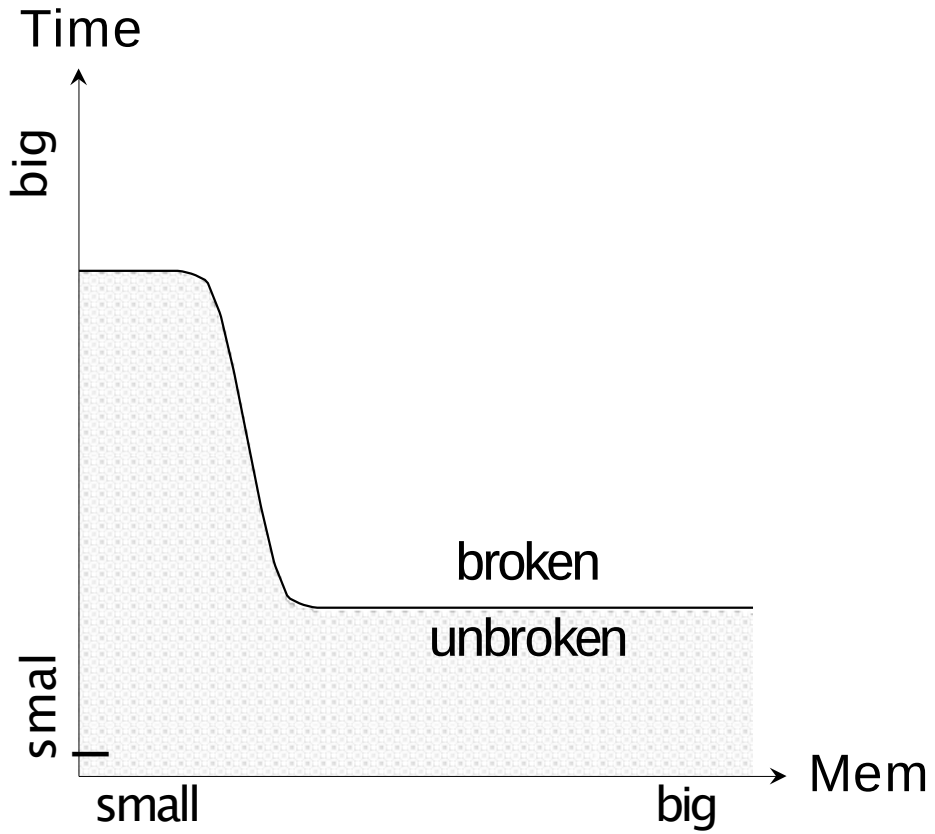
$\text{Mem} \approx 2^{\lambda/\log\lambda}$, $\text{Time} \approx 2^{\lambda/\log\lambda}$

Example: Time-Memory Trade-off for LPN



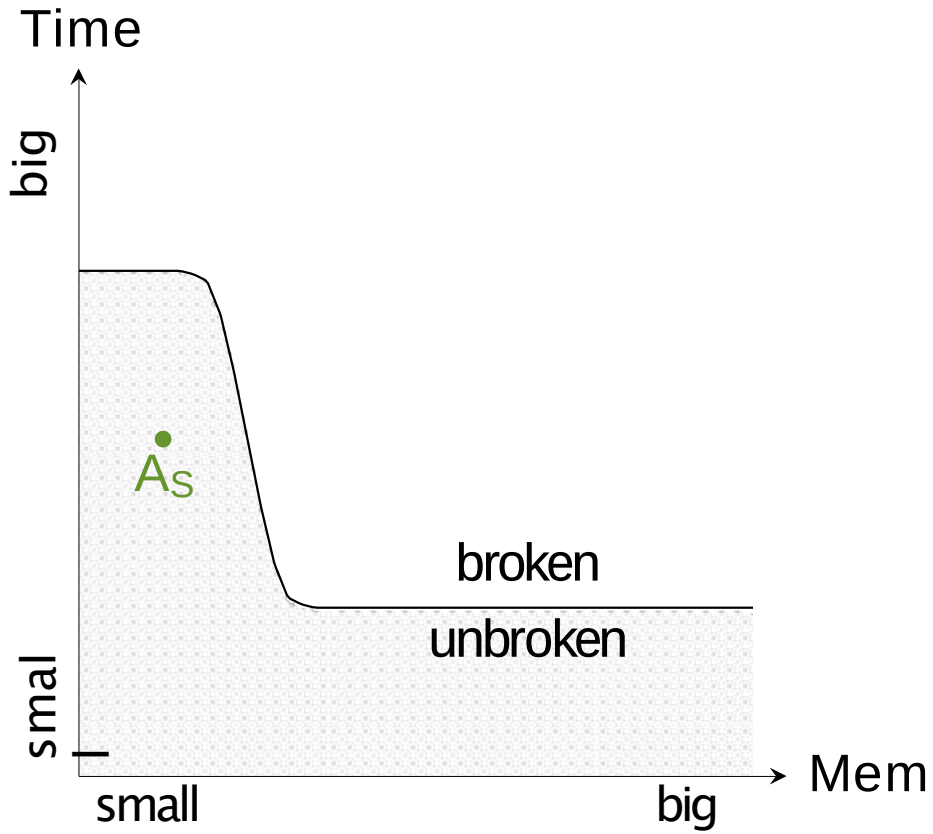
Time-memory trade-off plot for LPN

Example: Time-Memory Trade-off for LPN

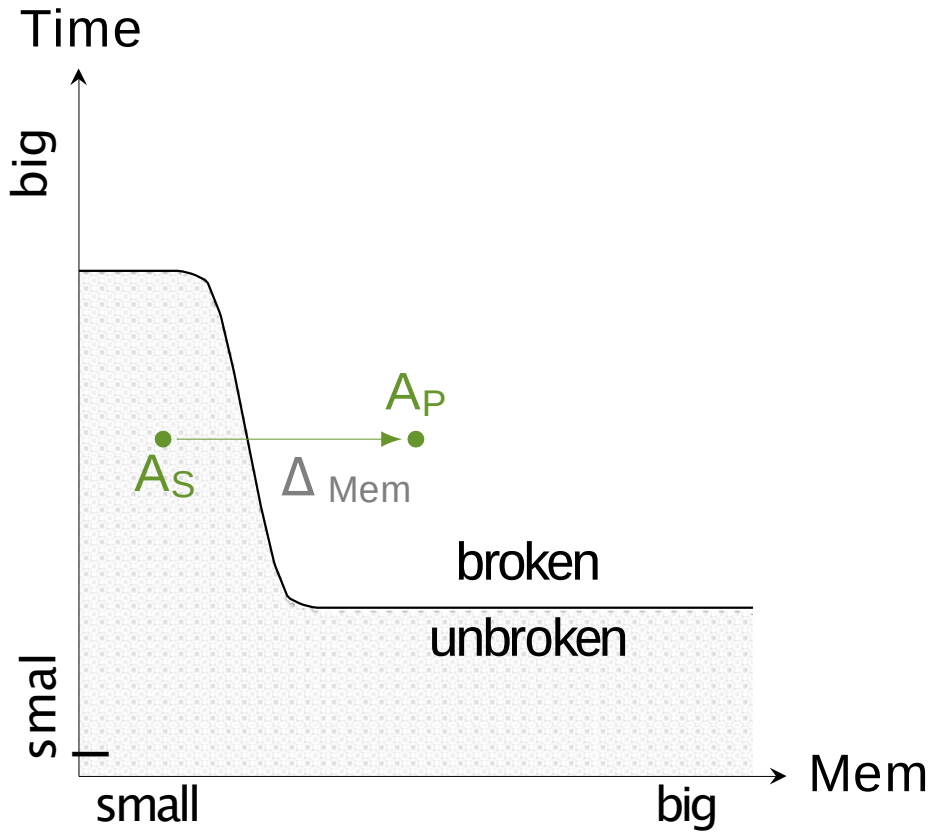


Time-memory trade-off plot for LPN [EKM, 11:00 Lotte Lehman Hall]

Memory-Tight Reductions

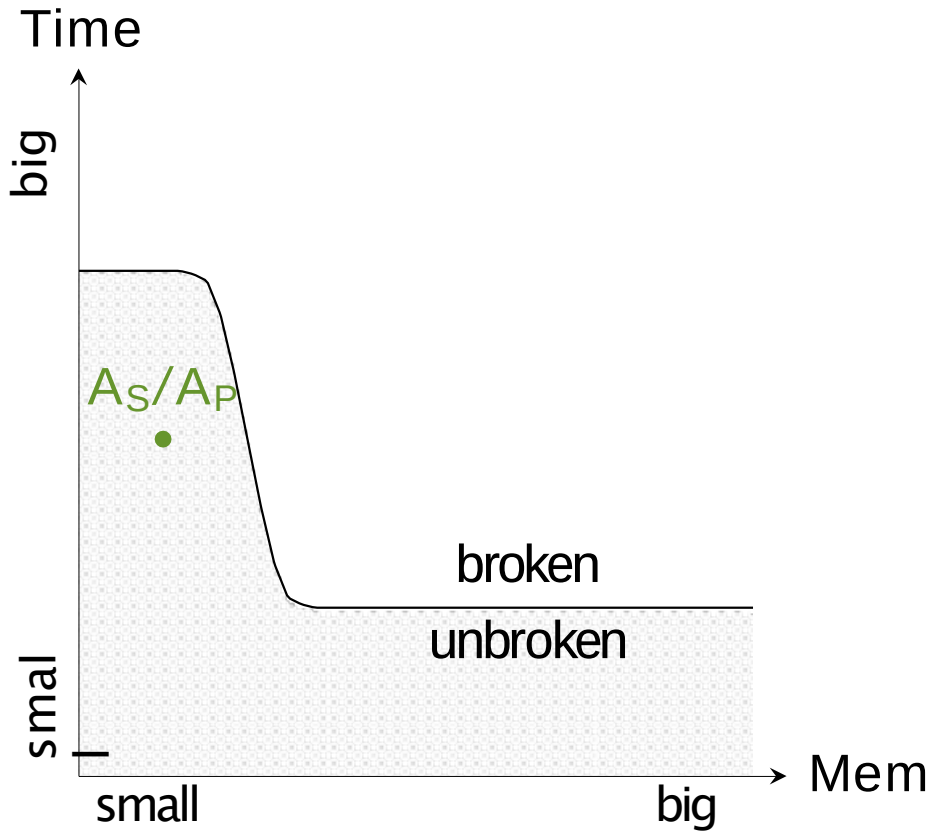


Memory-Tight Reductions



(Time-)Tight but not memory-tight reduction

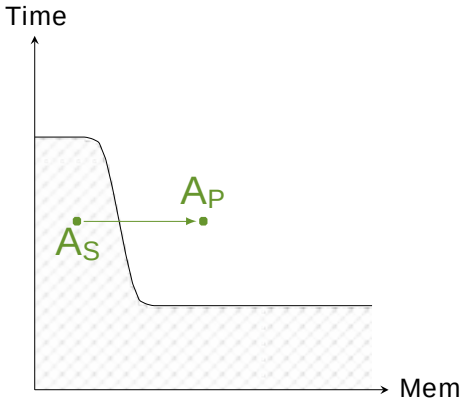
Memory-Tight Reductions

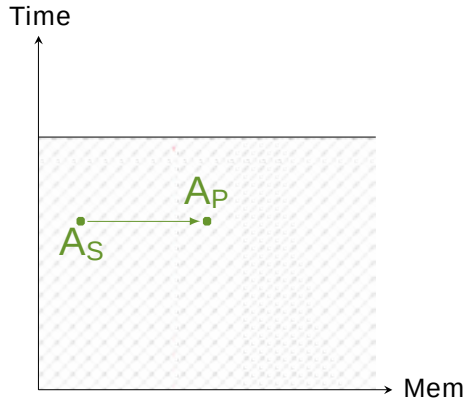


(Time-)Tight and memory-tight reduction

$\mathrm{Time}(A_P) \approx \mathrm{Time}(A_S), \mathrm{Mem}(A_P) \approx \mathrm{Mem}(A_S)$

Memory-Sensitive Problems





Memory-sensitive:

- LPN
- Shortest Vector Problem
- 3 collision resistance
- DLP in finite fields
- Factoring

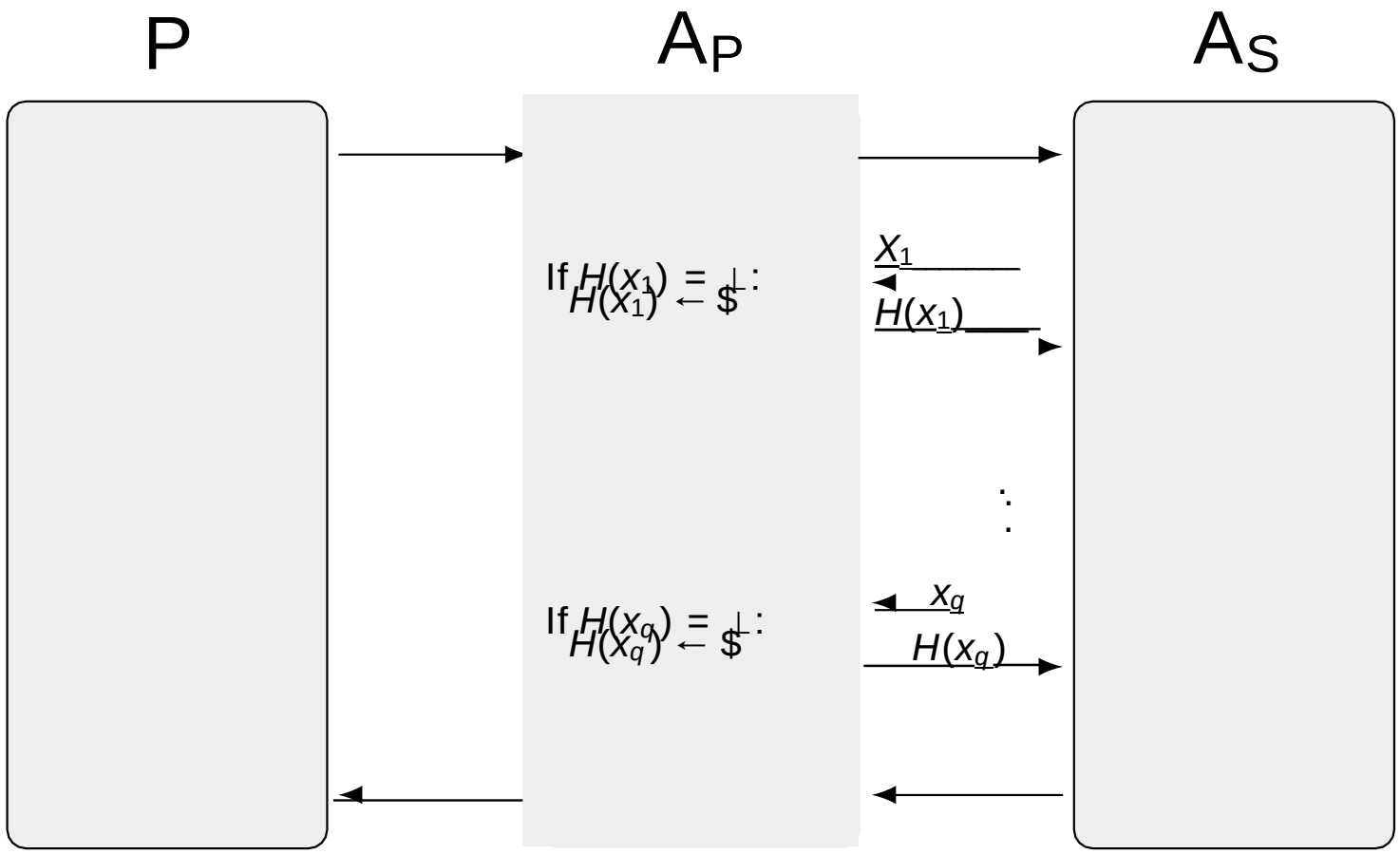
Not memory-sensitive:

- Collision resistance
- Preimage resistance
- DLP over elliptic curves

Might change in future

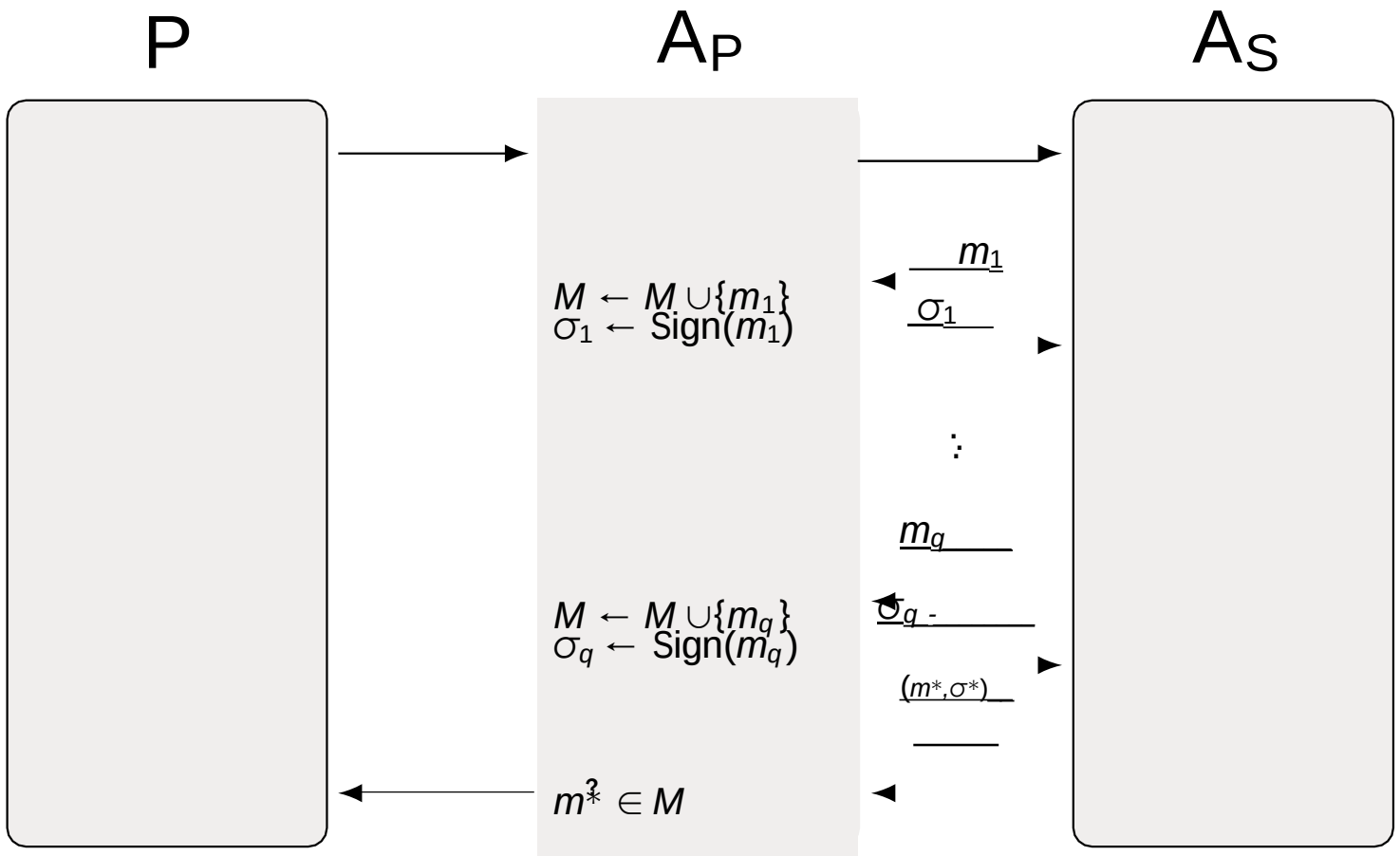
Typical Non-Memory-Tight Reductions

Example 1: Random Oracle Simulation



Worst case: $\mathsf{Mem}(A_P) \approx \mathsf{Time}(A_S)$, while $\mathsf{Mem}(A_S)$ small

Example 2: Unforgeability of Signatures



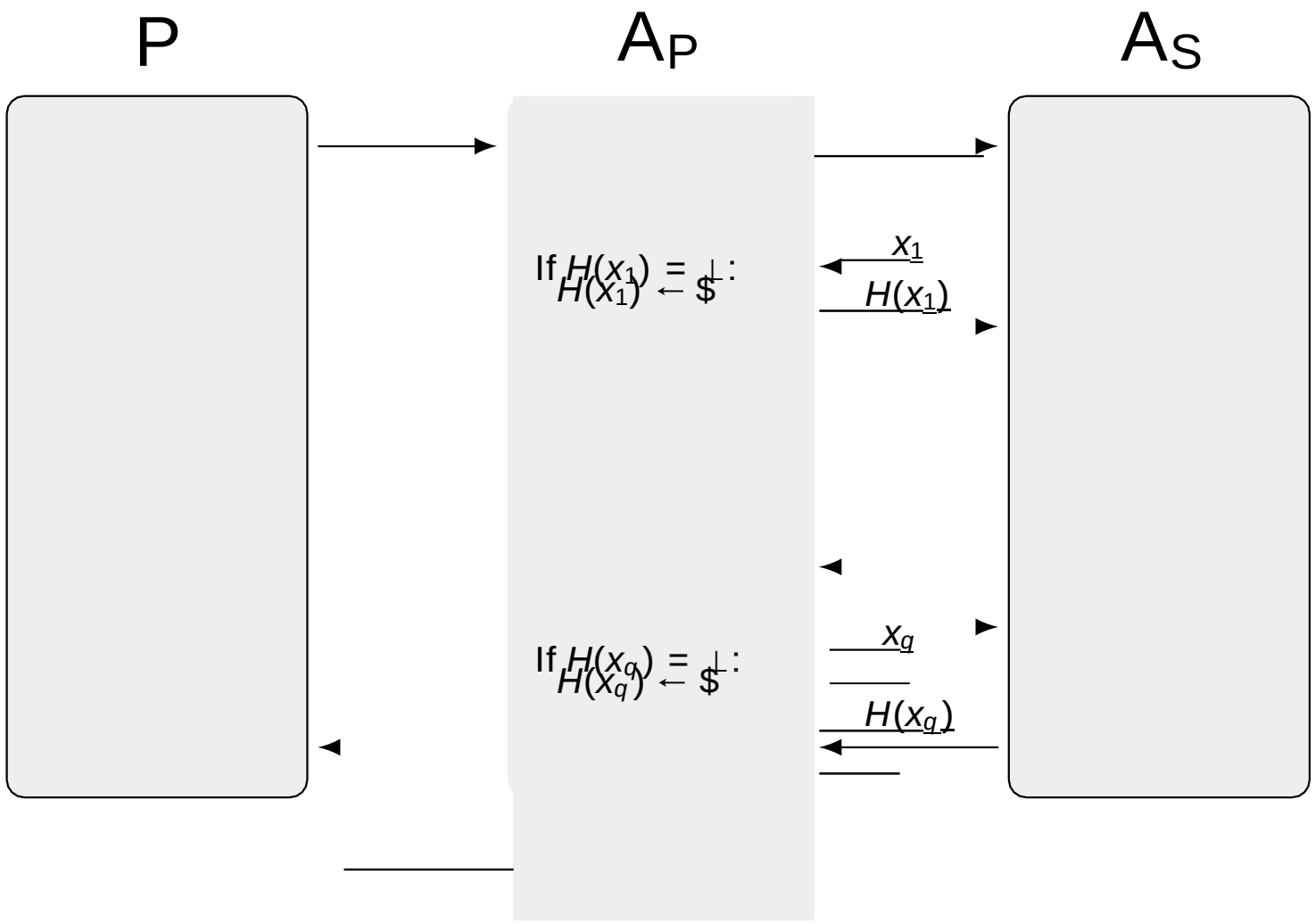
Worst case: $\mathrm{Mem}(A_P) \approx \mathrm{Time}(A_S)$, while $\mathrm{Mem}(A_S)$ small

Recap

- Currently memory often ignored in reductions
- Many existing reductions not memory-tight
  - Worst case $\mathsf{Mem}(A_P) \approx \mathsf{Time}(A_S)$ while $\mathsf{Mem}(A_S)$ small
- Particularly problematic for memory-sensitive problems

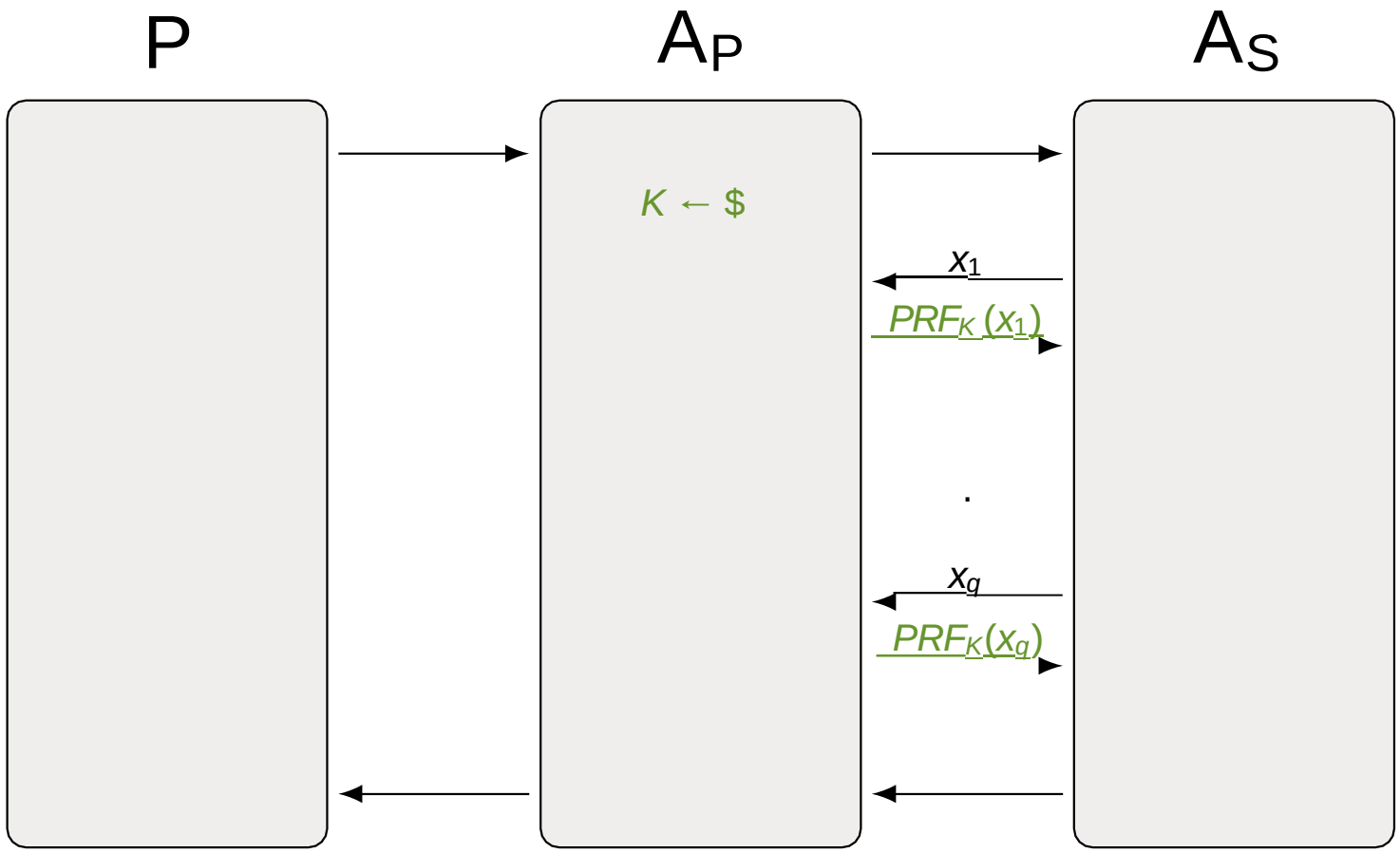
Achieving Memory-Tightness

Example 1: Random Oracle Simulation



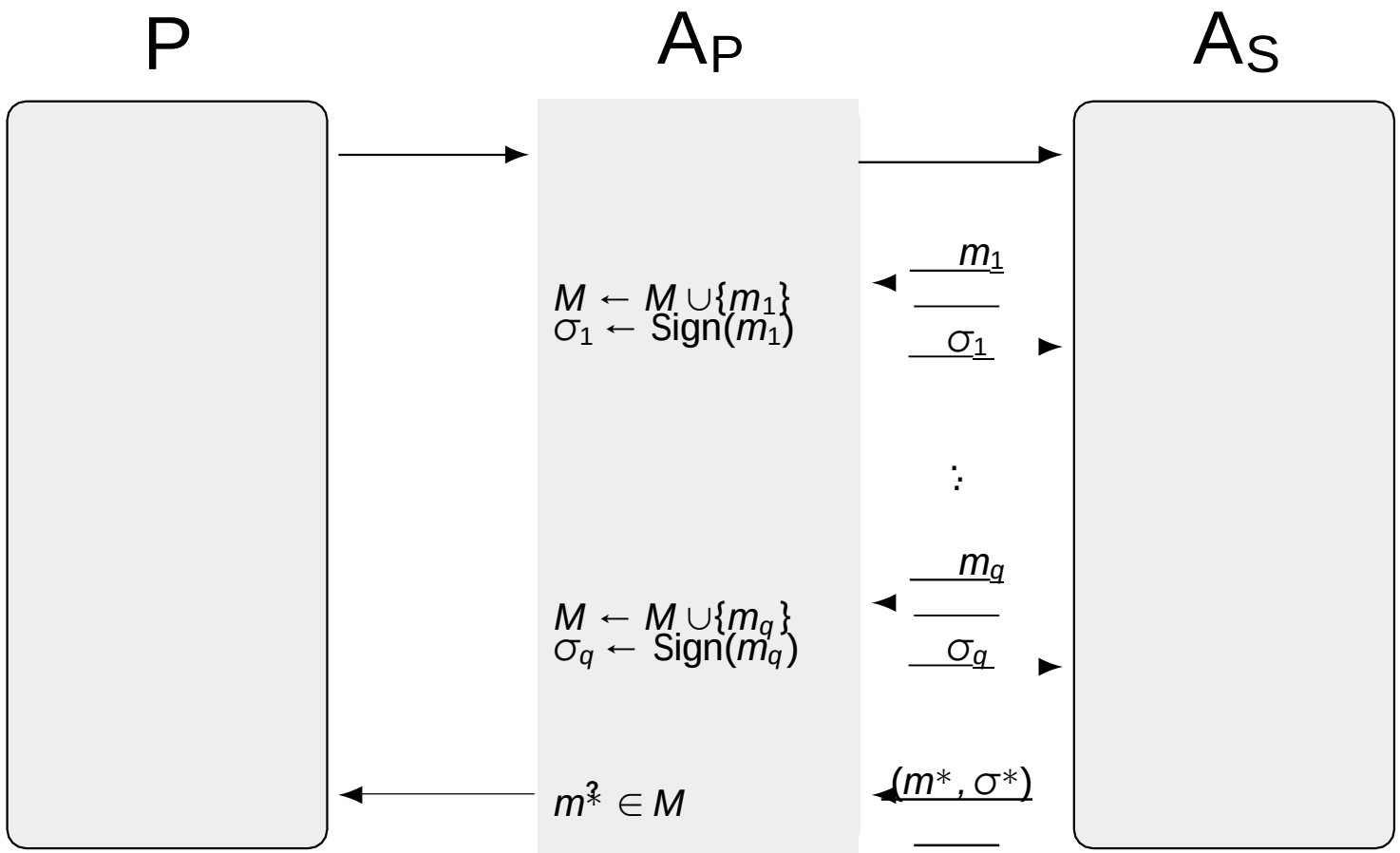
RO simulation via lazy sampling

Example 1: Random Oracle Simulation



Memory efficient RO simulation [Bernstein 2011]

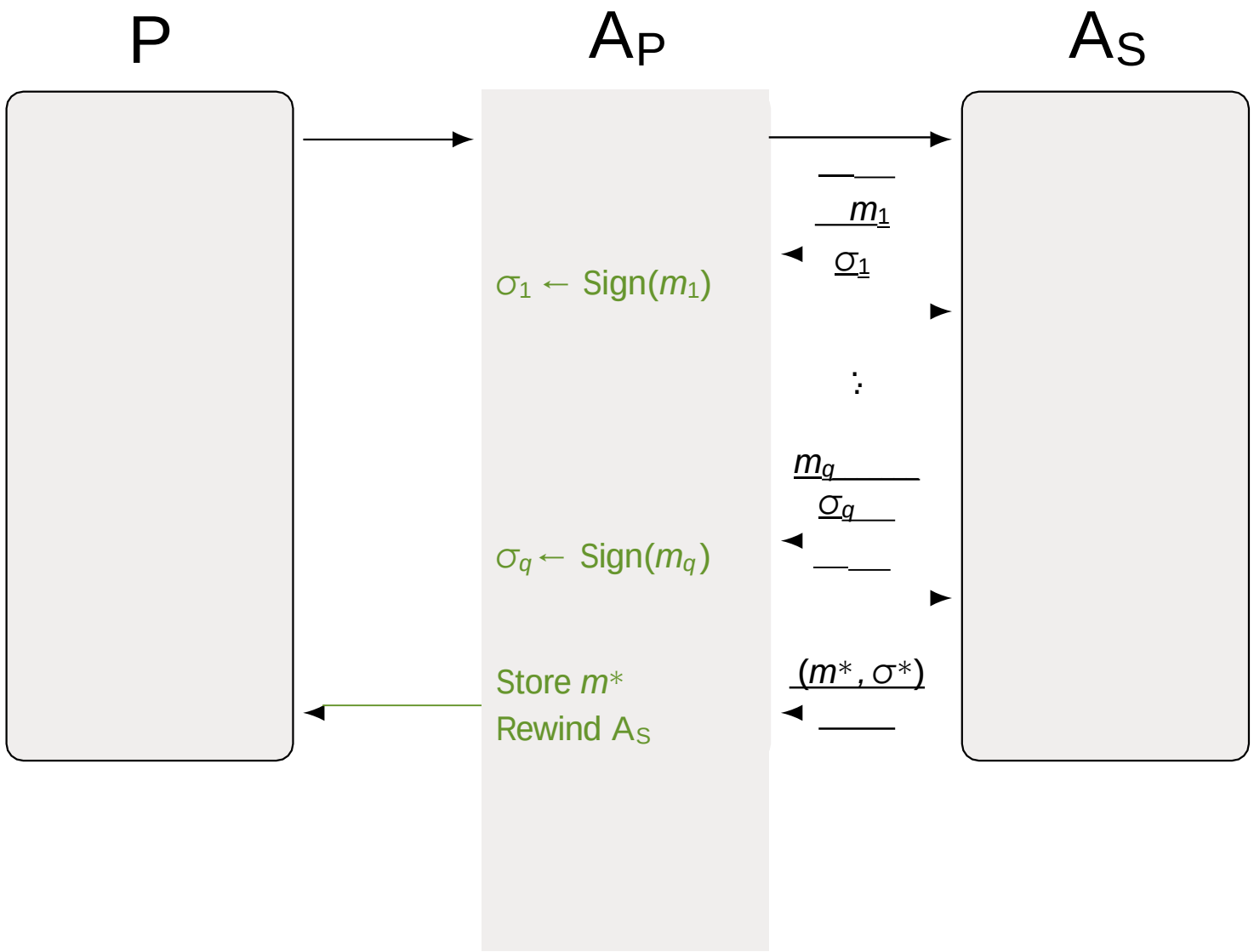
Example 2: Unforgeability of Signatures



Usual simulation of unforgeability game

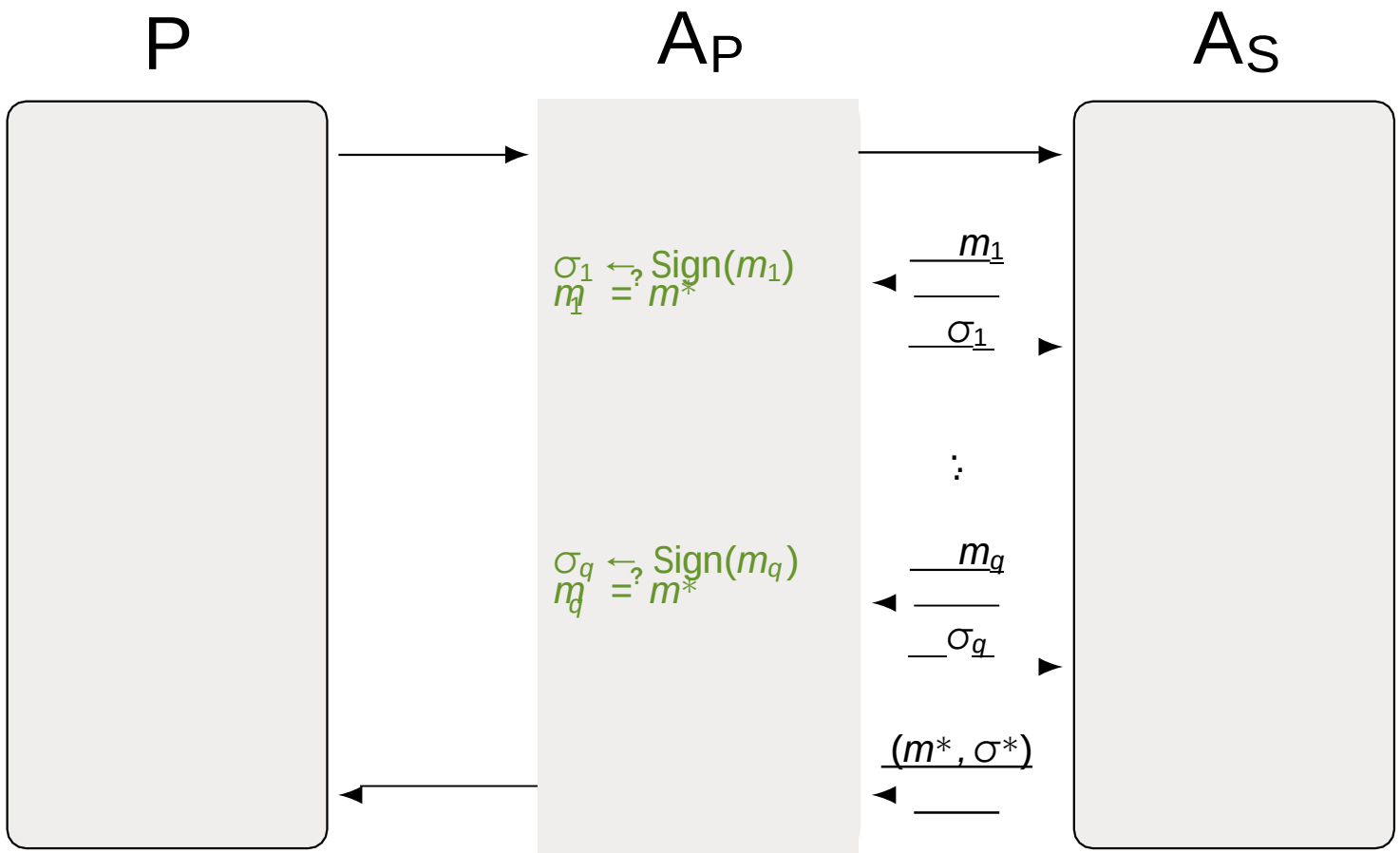
Benedikt Auerbach: Memory-Tight Reductions

Example 2: Unforgeability of Signatures



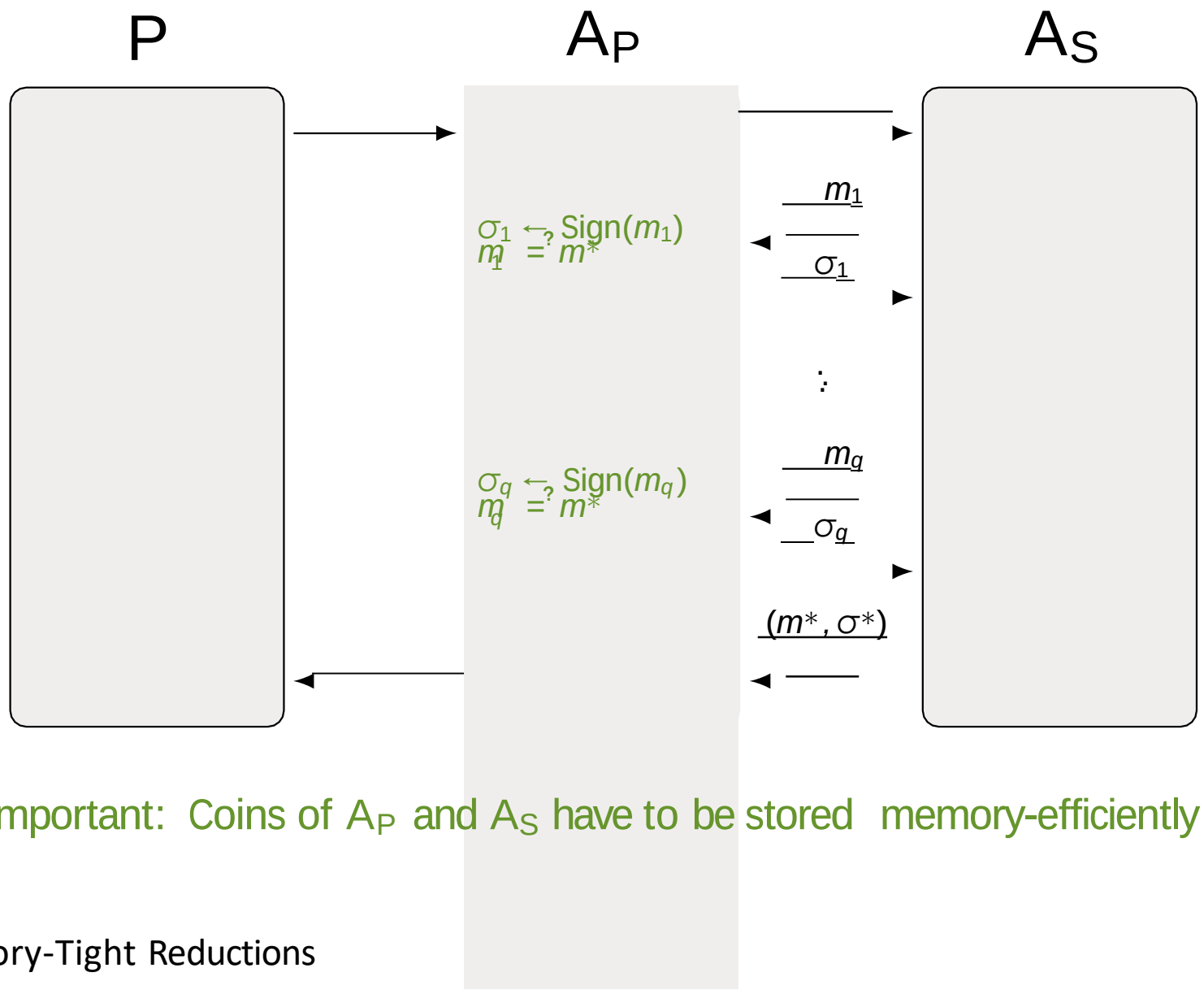
Memory-efficient simulation of unforgeability game

Example 2: Unforgeability of Signatures



Memory-efficient simulation of unforgeability game

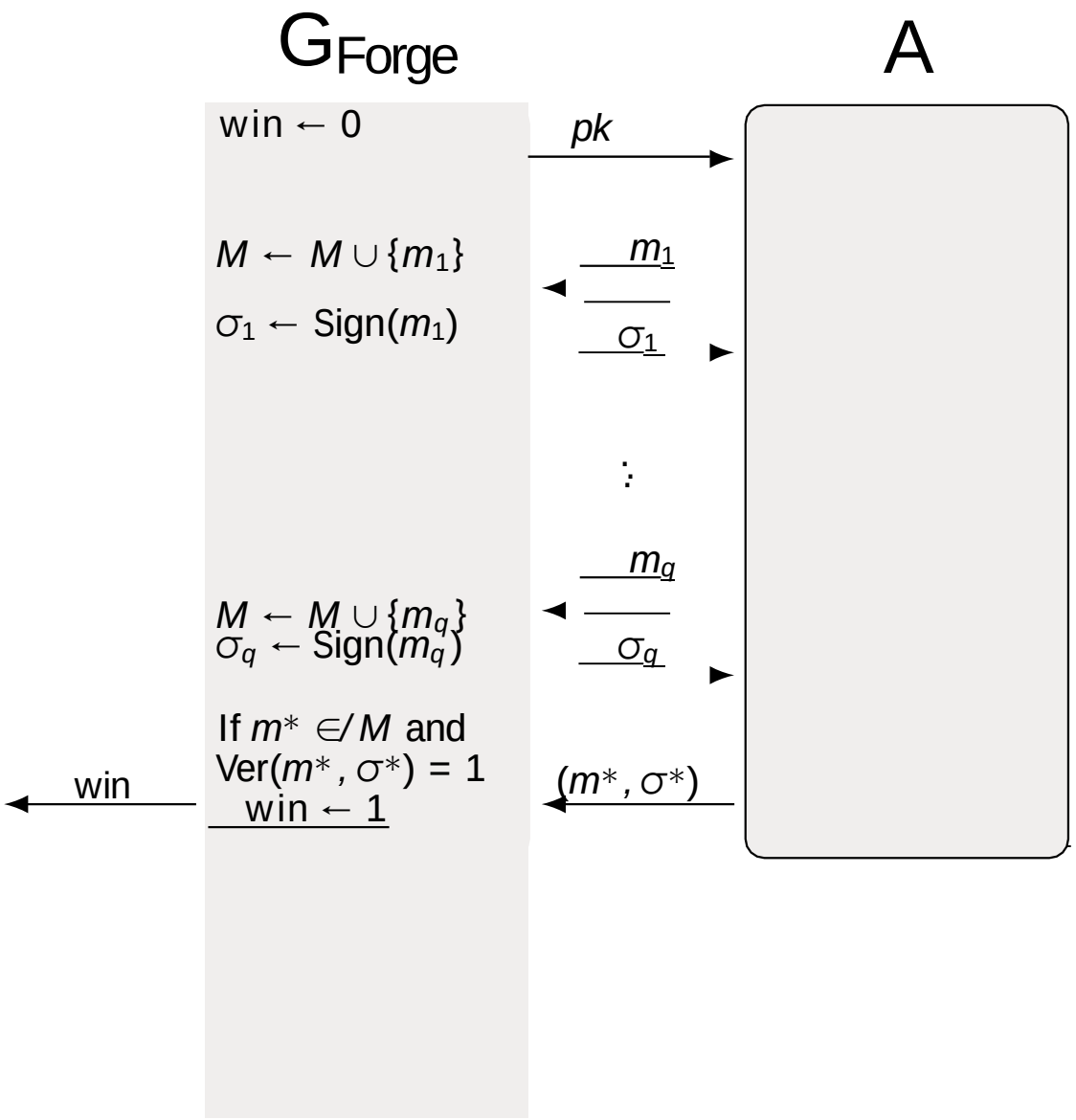
Example 2: Unforgeability of Signatures



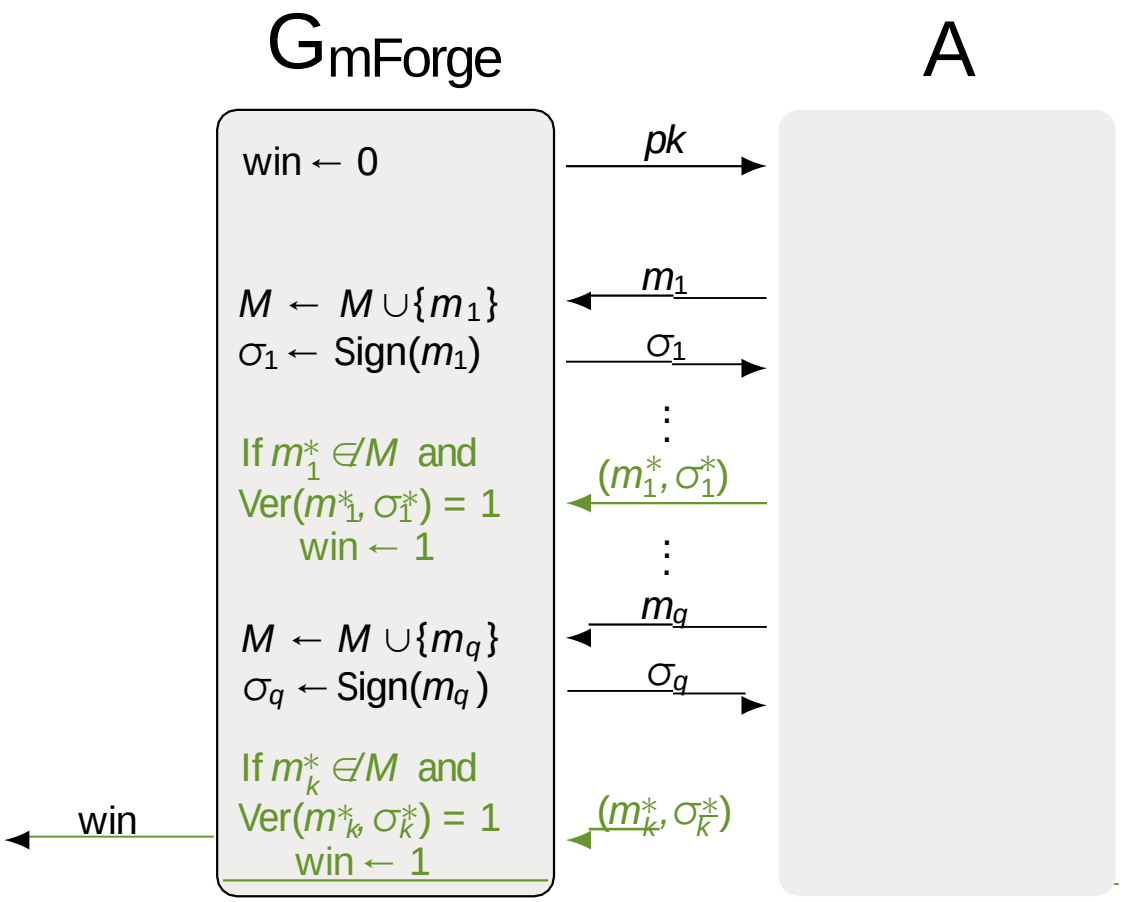
Important: Coins of A_P and A_S have to be stored memory-efficiently

Lower Bounds

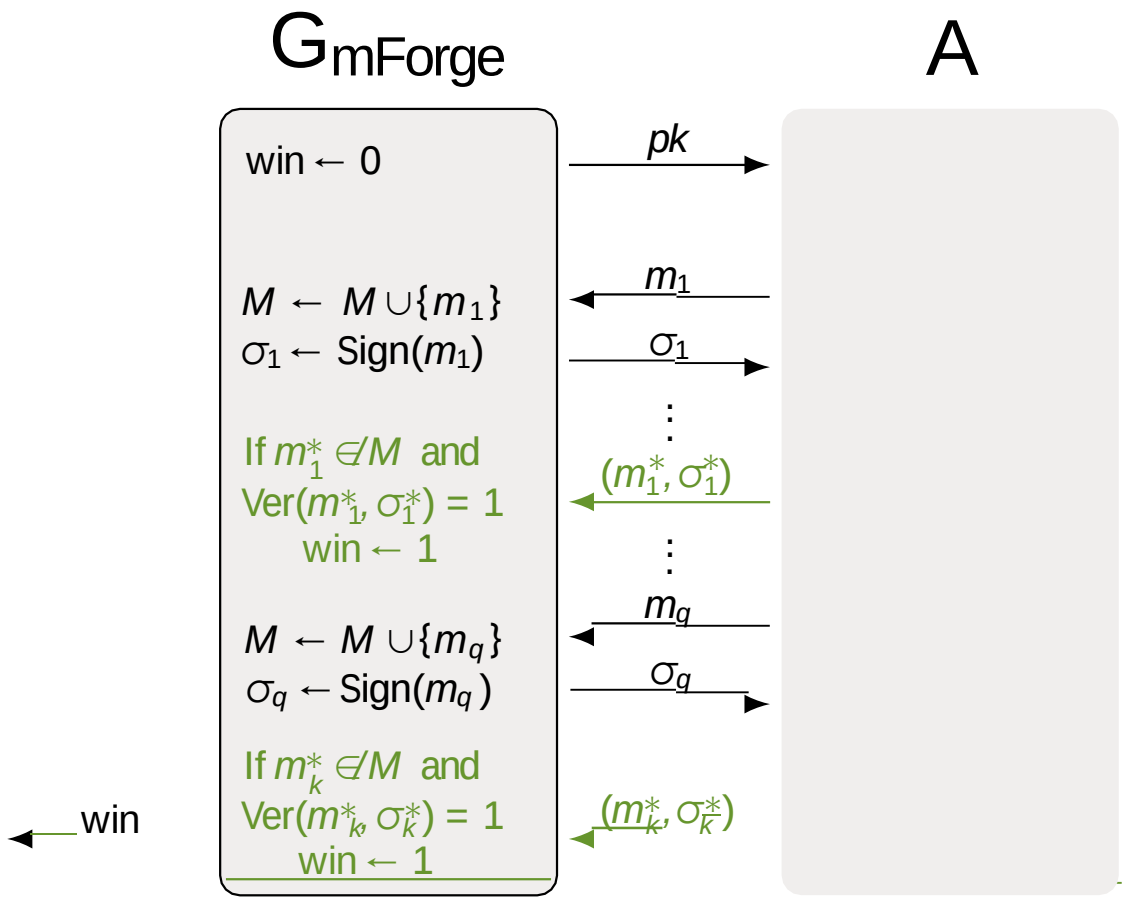
Unforgeability and Multi-Unforgeability



Unforgeability and Multi-Unforgeability

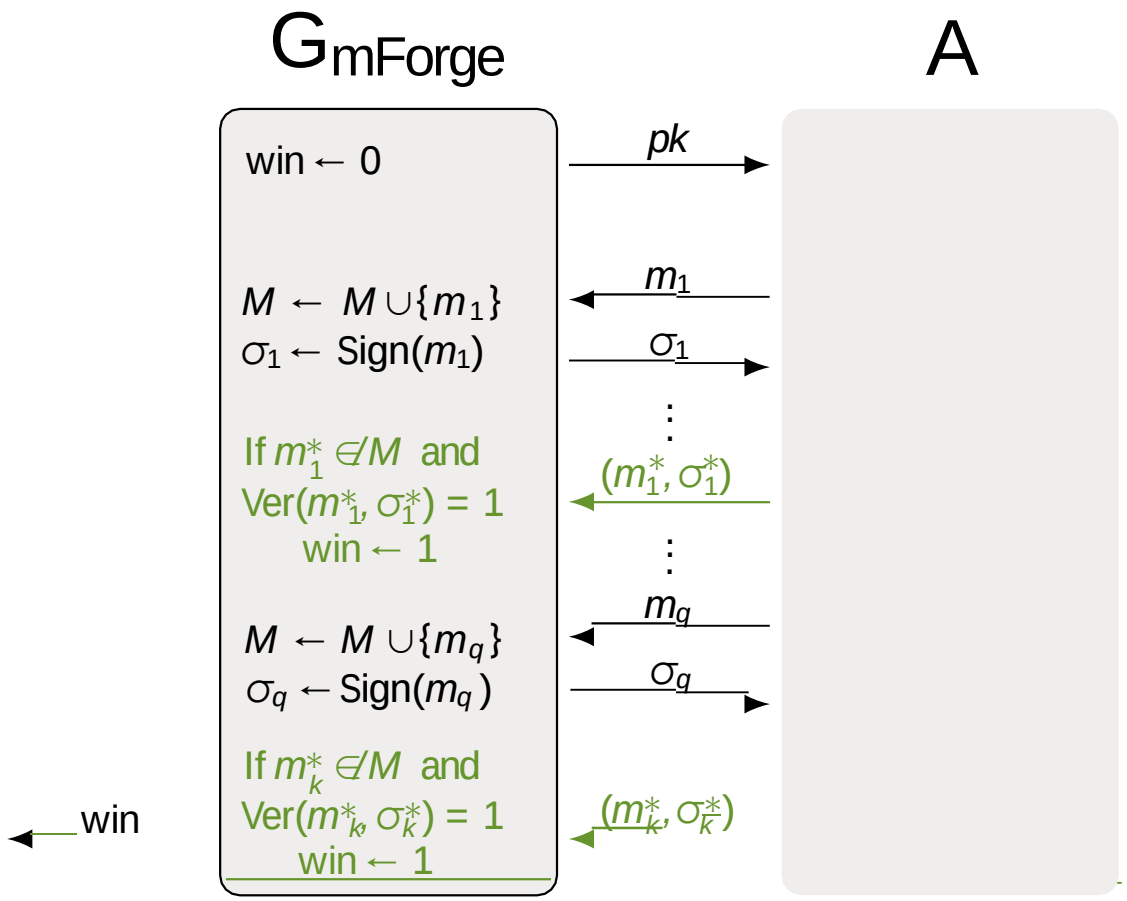


Unforgeability and Multi-Unforgeability



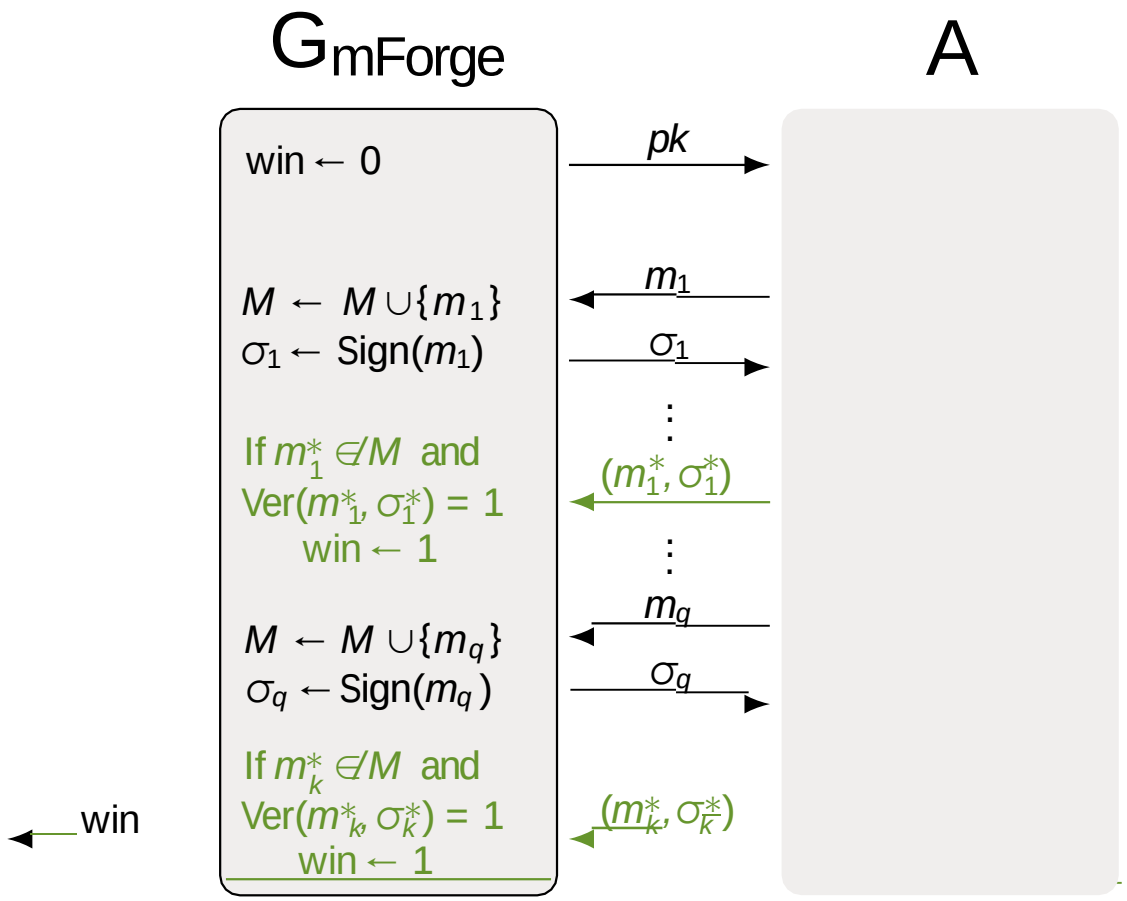
Tight but not memory-tight reduction (store queries)

Unforgeability and Multi-Unforgeability



Memory-tight reduction with lower success probability (guess forgery)

Unforgeability and Multi-Unforgeability



Memory-tight reduction with higher running time (rewind adversary)

Unforgeability and Multi-Unforgeability

Reductions from mForge to Forge (for q adversarial queries)

	Time	Succ	Mem
store queries	1	1	q
guess forgery	1	$1/q$	1
rewind adversary	q	1	1

Lower Bounds

Theorem

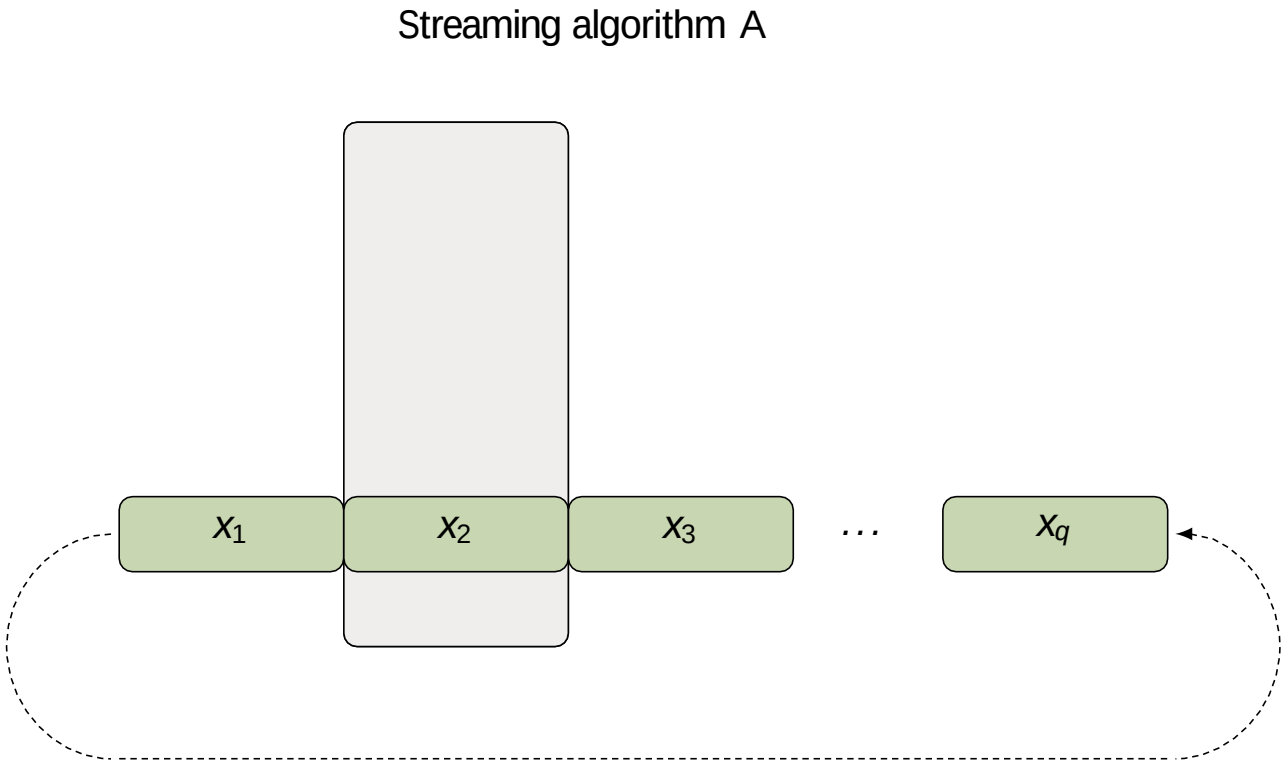
A certain class of black box reductions from mForge to Forge can not be simultaneously tight and memory-tight.

- Proof uses techniques from streaming algorithms
- Similar results for multi-collision to collision resistance

Lower Bounds

Theorem

A certain class of black box reductions from mForge to Forge can not be simultaneously tight and memory-tight.



Conclusions and Future Work

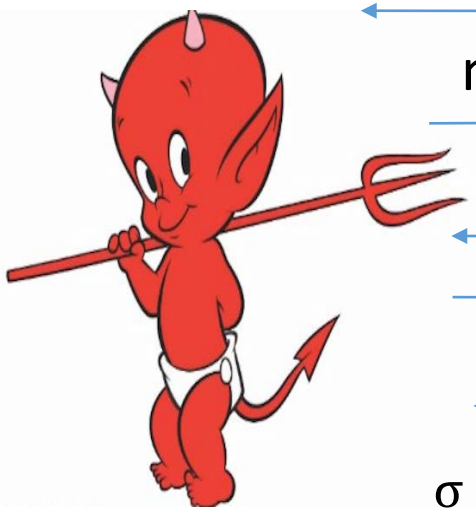
- Memory usage is ignored but affects security.
- Many reductions are easily fixed...
- ... but some seem inherently loose, including some widely used implicitly.

Future work:

- Give memory-tight reductions for some constructions (e.g. Hashed ElGamal).
- Prove lower bounds in less restrictive models.

ia.cr/2017/675

Signature Experiment ($\mathrm{Sig} - \mathrm{forge}_{\mathrm{A},\Pi}(\mathrm{n})$)



Public Key: pk

m_1

$\sigma_1 = \mathbf{Sign}_{sk}(\boldsymbol{m_1})$

m_2

$\sigma_2 = \mathbf{Sign}_{sk}(\boldsymbol{m_2})$

$\sigma, m \notin \mathfrak{Q} = \{m_1, m_2 \ldots\}$

$\mathrm{Sig} - \mathrm{forge}_{\mathrm{A},\Pi}(\mathrm{n}) = \mathrm{Vrfy}_{\mathrm{pk}}(m, \sigma)$

(pk,sk) = Gen(.)

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$

$$\Pr\left[\mathrm{Sig} - \mathrm{forge}_{\mathrm{A},\Pi}(\mathrm{n}) = 1\right] \leq \mu(n)$$

Signature Experiment ($\mathrm{Sig} - \mathrm{forge}_{A,\Pi}(n)$)

Formally, let $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ denote the signature scheme, call the experiment $\mathrm{Sig} - \mathrm{forge}_{A,\Pi}(n)$

We say that Π *is existentially unforgeable under an adaptive chosen message attack* (or just *secure*) if for all PPT adversaries A, there *is a* negligible function μ such that

$$\Pr[\mathrm{Sig} - \mathrm{forge}_{A,\Pi}(n) = 1] \leq \mu(n)$$

Existential Unforgeability

- **Limitation**: Does not prevent replay attacks
  - $\sigma \leftarrow \mathrm{Sign}_{\mathrm{sk}}(\text{"}Pay\ Bob\ \$50\text{"}, R)$
  - If this is a problem then you can include timestamp in signature
- Unforgeability: does rule out the possibility attacker modifies a signature
- Plain RSA signatures are malleable (does not satisfy our security notion)
- **Remark:** By design signatures cannot hide all information about message *m*
  - Public Verification → Attacker can easily distinguish between a signature for m_1 and m_2

Plain RSA Signatures

- Plain RSA
- Public Key (pk): N = pq, e such that $\mathrm{GCD}\left(\mathrm{e}, \phi(N)\right) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Secret Key (sk): N, d such that $\mathrm{ed} = 1 \bmod \phi(N)$

$$\mathbf{Sign}_{\boldsymbol{sk}}(\boldsymbol{m}) = \boldsymbol{m}^{\boldsymbol{d}} \boldsymbol{\bmod N}$$

$$\mathbf{Vrfy}_{\boldsymbol{pk}}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if\ m} = [\sigma^{\boldsymbol{e}} \boldsymbol{\bmod N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

- Verification Works because

$$\left[\mathbf{Sign}_{\boldsymbol{sk}}(\boldsymbol{m})^{e} \bmod \mathrm{N}\right] = \left[m^{ed} \bmod \mathrm{N}\right] = \left[m^{[ed \bmod \boldsymbol{\phi}(\boldsymbol{N})]} \bmod \mathrm{N}\right] = m$$

No Message Attack

- **Goal:** Generate a forgery using only the public key
  - No intercepted signatures required
- Public Key (pk): $\mathrm{N} = \mathrm{pq}$, e such that $\mathrm{GCD}\big(\mathrm{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Pick random $\sigma \in \mathbb{Z}_{\mathrm{N}}^{*}$
- Set $\boldsymbol{m} = [\sigma^{\boldsymbol{e}} \ \boldsymbol{mod}\ \boldsymbol{N}]$.
- Output (m, σ)

$$\mathbf{Vrfy}_{\boldsymbol{pk}}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if}\ \boldsymbol{m} = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

RSA-FDH

- **Full Domain Hash:** $H: \{0,1\}^* \rightarrow \mathbb{Z}_N$
- Given a message $m \in \{0,1\}^*$

$$\sigma = \mathrm{Sign}_{sk}(m) = H(m)^d \ mod \ N$$

Theorem 12.7: RSA-FDH is a secure signature scheme assuming that the RSA-Inversion problem is hard and H is modeled as a random oracle.

Remark: The range of H (e.g.,SHA3) may be shorter than $\mathbb{Z}_N$ **.**

Solution: Repeated application of H e.g., $H'(m) = H(1|\, m) \ \ldots H(k|m) \bmod \mathrm{N}$

RSA-FDH

- **Full Domain Hash:** $H: \{0,1\}^* \rightarrow \mathbb{Z}_N^*$
- Given a message $m \in \{0,1\}^*$

$$\sigma = \mathrm{Sign}_{sk}(m) = H(m)^d \ mod \ N$$

Theorem 12.7: RSA-FDH is a secure signature scheme assuming that the RSA-Inversion problem is hard and H is modeled as a random oracle.

Proof Sketch: Given an RSA-Inversion challenge $c = r^e \ mod \ N$ (r is unknown) we will simulate the signature attacker. WLOG assume attacker always queries $\mathrm{H}(\mathrm{m_i})$ before $\mathrm{Sign}_{sk}(\mathrm{m_i})$

1. Whenever the attacker queries $\mathrm{H}(\mathrm{m_i})$ we can pick a random $\mathrm{r_i} \in \mathbb{Z}_N^*$ and program $\mathrm{H}(\mathrm{m_i}) = \mathrm{r_i}^e \ mod \ N$ so that $\mathrm{H}(\mathrm{m_i})^d = \mathrm{r_i} \ mod \ N$.
2. Whenever the attacker queries $\mathrm{Sign}_{sk}(\mathrm{m_i})$ we can simply return $\mathrm{r_i} \in \mathbb{Z}_N^*$
3. Exception: Pick a random query index $j \leq q_{hash}$ and program $\mathrm{H}(\mathrm{m_i}) = \mathrm{c} \times \mathrm{r_i}^e \ mod \ N$ instead of $\mathrm{H}(\mathrm{m_i}) = \mathrm{r_i}^e \ mod \ N$
4. If the attacker forges a signature σ for $\mathrm{m_i}$ we can win the RSA-Inversion game by computing $r = \sigma \times \mathrm{r_i}^{-1}$ since $\sigma = \mathrm{Sign}_{sk}(m) = (\mathrm{c} \times \mathrm{r_i}^e)^d = (r \times \mathrm{r_i})^{ed} = r\mathrm{r_i} \ mod \ N$.

Analysis: If signature forgery attacker wins with probability f(n) we win RSA-inversion game with probability f(n)/q_{hash} where q_{hash} is the number of queries to the random oracle.

RSA-FDH

- **Full Domain Hash:** $H: \{0,1\}^* \rightarrow \mathbb{Z}_N^*$
- Given a message $m \in \{0,1\}^*$

$$\sigma = \mathrm{Sign}_{sk}(m) = H(m)^d \ mod \ N$$

Theorem 12.7 (Concrete): Suppose that any attacker running in time at most $t'(n)$ wins RSA-Inversion game with probability at most $\varepsilon'(n)$ then the RSA-FDH is $\big(t(n), q_H, q_{Sig}, \varepsilon(n)\big)$-secure i.e., any attacker running in time $t(n) = t'(n) - \big(q_H + q_{Sig} + 1\big)O\big(poly(n)\big)$ and making at most q_H (resp. q_{Sig}) queries to the random oracle (resp. signature oracle) wins with probability at most $\varepsilon(n) \leq 4q_{Sig}\varepsilon'(n)$

RSA-FDH

Theorem 12.7 (Concrete): Suppose that any attacker running in time at most $t'(n)$ wins RSA-Inversion game with probability at most $\varepsilon'(n)$ then the RSA-FDH is $\left(t(n), q_H, q_{Sig}, \varepsilon(n)\right)$-secure i.e., any attacker running in time $t(n) = t'(n) - \left(q_H + q_{Sig} + 1\right)O\left(poly(n)\right)$ and making at most q_H (resp. q_{Sig}) queries to the random oracle (resp. signature oracle) wins with probability at most $\varepsilon(n) \leq 4q_{Sig}\varepsilon'(n)$

Proof Idea: Whenever the attacker queries $\mathrm{H}(\mathrm{m_i})$ we can pick a random $\mathrm{r_i} \in \mathbb{Z}_N^*$ and flip a biased coin $\Pr[\text{heads}] = \left(1 - \frac{1}{1+q_{Sig}}\right)$

1. Heads: program $\mathrm{H}(\mathrm{m_i}) = \mathrm{r_i}^e \; mod \; N$ so that $\mathrm{H}(\mathrm{m_i})^d = \mathrm{r_i} \, mod \; N$.
2. Tails: and program $\mathrm{H}(\mathrm{m_i}) = \mathrm{c} \times \mathrm{r_i}^e \; mod \; N$
    1. If attacker queries signing oracle on this message we will need to abort

$$\Pr[no \text{ abort}] = \Pr[\text{heads}]^{q_{Sig}} = \left(1 - \frac{1}{1 + q_{Sig}}\right)^{q_{Sig}} \approx \frac{1}{e}$$

RSA-FDH

Theorem 12.7 (Concrete): Suppose that any attacker running in time at most $t'(n)$ wins RSA-Inversion game with probability at most $\varepsilon'(n)$ then the RSA-FDH is $\left(t(n), q_H, q_{Sig}, \varepsilon(n)\right)$-secure i.e., any attacker running in time $t(n) = t'(n) - \left(q_H + q_{Sig} + 1\right)O\left(poly(n)\right)$ and making at most q_H (resp. q_{Sig}) queries to the random oracle (resp. signature oracle) wins with probability at most $\varepsilon(n) \leq 4q_{Sig}\varepsilon'(n)$

Proof Idea: Whenever the attacker queries $\mathrm{H}(\mathrm{m_i})$ we pick a random $\mathrm{r_i} \in \mathbb{Z}_N^*$ and flip a biased coin $\Pr[\text{heads}] = \left(1 - \frac{1}{1+q_{Sig}}\right)$

1. Heads: program $\mathrm{H}(\mathrm{m_i}) = \mathrm{r_i}^e \; mod \; N$ so that $\mathrm{H}(\mathrm{m_i})^d = \mathrm{r_i} \, mod \; N$.
2. Tails: and program $\mathrm{H}(\mathrm{m_i}) = \mathrm{c} \times \mathrm{r_i}^e \; mod \; N$
    1. If attacker queries signing oracle on this message we will need to abort

$$\Pr[no \text{ abort}] = \Pr[\text{heads}]^{q_{Sig}} = \left(1 - \frac{1}{1 + q_{Sig}}\right)^{q_{Sig}} \approx \frac{1}{e}$$

Reducing Memory Usage in Reduction

Prior reduction is not memory-tight since we need to remember that $\mathrm{H}(\mathrm{m_i}) = \mathrm{r_i}^e \ mod\ N$ for each query.

Solution: Set $\mathrm{r_i} \ = H \ (K, \mathrm{m_i})$ where K is secret key used in the reduction. Program $\mathrm{H}(\mathrm{m_i})$ as before.

flip a biased coin $\Pr[\mathrm{heads}] = \left(1 - \frac{1}{1+q_{Sig}}\right)$

1. Heads: program $\mathrm{H}(\mathrm{m_i}) = \mathrm{r_i}^e \ mod\ N$ so that $\mathrm{H}(\mathrm{m_i})^d = \mathrm{r_i} \ mod\ N$.
2. Tails: and program $\mathrm{H}(\mathrm{m_i}) = \mathrm{c} \times \mathrm{r_i}^e \ mod\ N$
    1. If attacker queries signing oracle on this message we will need to abort

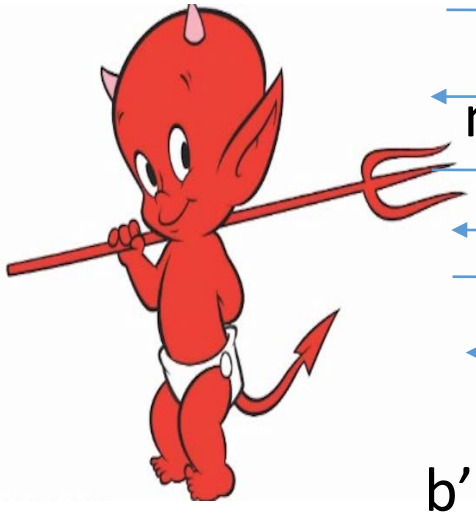
$$\Pr[no\ \mathrm{abort}] = \Pr[\mathrm{heads}]^{q_{Sig}} = \left(1 - \frac{1}{1+q_{Sig}}\right)^{q_{Sig}} \approx \frac{1}{e}$$

Idea 1: If the attacker does not query $H \ (K, \dots)$ then the reduction is unchanged.

Idea 2: If the attacker forges a signature σ for $\mathrm{m_i}$ we will hope that we programmed $\mathrm{H}(\mathrm{m_i}) = \mathrm{c} \times \mathrm{r_i}^e \ mod\ N$ (tails) and compute $\mathrm{r_i} \ = H \ (K, \mathrm{m_i})$

$r = \sigma \times \mathrm{r_i}^{-1}$ since $\sigma = \mathrm{Sign}_{sk}(m) = (\mathrm{c} \times \mathrm{r_i}^e)^d = (r \times \mathrm{r_i})^{ed} = r\mathrm{r_i} \ mod\ N.$

CPA-Security Game (Single Message Version)



m_0, m_1

$\langle r, F_k(r) \oplus m_b \rangle$

m_2

$\langle r_2, F_k(r_2) \oplus m_2 \rangle$

m_3

$\langle r_3, F_k(r_3) \oplus m_3 \rangle$

⋮

b’

Random bit b

$\mathbf{K} \leftarrow \mathbf{Gen}(\mathbf{1}^{n})$

$(t(n), q(n), \varepsilon(n))$-secure if any attacker A running in time t and making at most q queries wins with probability at most $\frac{1}{2} + \varepsilon(n)$

Recall: Week 1 Reduction

$$\text{Enc}_k(m) = \langle r, F_k(r) \oplus m \rangle$$

$$\text{Dec}_k(\langle r, s \rangle) = F_k(r) \oplus s$$

For any attacker A running in time $t(n)$ and making at most q(n) encryption queries we have

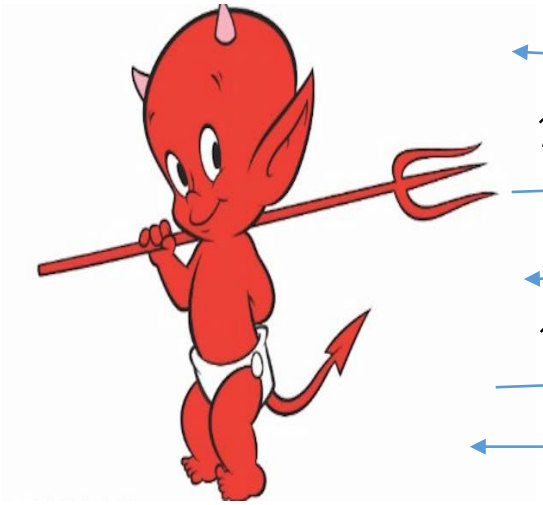
$$\Pr\left[PrivK^{cpa1}_{A,\Pi} = 1\right] \leq \frac{1}{2} + \frac{q(n)}{2^n} + \mu(n, t'(n), q(n))$$

PRF Security

$t'(n) \approx t(n)$

Probability of Nonce Collision with initial challenge

CPA-Security Game (Left-Right)



m_0^1, m_1^1

$\langle r, F_k(r) \oplus m_b^1 \rangle$

m_0^2, m_1^2

$\langle r_2, F_k(r_2) \oplus m_b^2 \rangle$

m_0^3, m_1^3

$\langle r_3, F_k(r_3) \oplus m_b^3 \rangle$

⋮ b'

Random bit b

$\mathbf{K} \leftarrow \mathbf{Gen}(\mathbf{1}^n)$

$\big(t(n), q(n), \varepsilon(n)\big)$-*secure if any attacker A running in time t and making at most q encryption queries wins with probability at most* $\frac{1}{2} + \varepsilon(n)$

Example: Left-Right Security

$$\text{Enc}_k(m) = \langle r, F_k(r) \oplus m \rangle$$

$$\text{Dec}_k(\langle r, s \rangle) = F_k(r) \oplus s$$

For any attacker A running in time $t(n)$ and making at most q(n) encryption queries we have

$$\Pr\left[PrivK_{A,\Pi}^{cpaLR} = 1\right] \leq \frac{1}{2} + \frac{\binom{q(n)}{2}}{2^n} + \mu(n, t'(n), q(n))$$

$t'(n) \approx t(n)$

PRF Security

Probability there exists a nonce collision

Example: Left-Right Security

Question: Suppose the attacker is space bounded and that $S \ll q(n)$ so that the attacker cannot store all of the random nonces. Can we prove a tighter security bound?

Fo
e

$$\Pr\left[PrivK_{A,\Pi}^{cpaLR} = 1\right] \leq \frac{1}{2} + \frac{\binom{q(n)}{2}}{2^n} + \mu(n, t'(n), q(n))$$

$t'(n) \approx t(n)$

PRF Security

Probability there exists a nonce collision

On the Streaming Indistinguishability of a Random Permutation and a Random Function

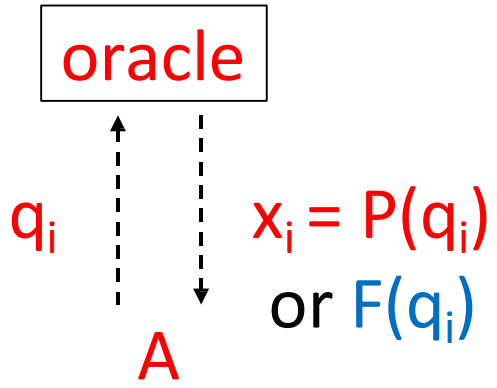
Itai Dinur

Ben-Gurion University

Eurocrypt 2020

"Switching Lemma" for Random Permutation\Function

- Classical problem: adversary A tries to **distinguish** a **random permutation** P:[N]->[N] from **random function** F:[N]->[N] with Q queries
- "Switching Lemma": A has **advantage** bounded by $O(Q^2/N)$
  - $| \Pr[A^{P(\cdot)} = 1] - \Pr[A^{F(\cdot)} = 1] | \in O(Q^2/N)$
- Widely used to establish concrete security of cryptosystems up to **birthday bound** of $Q = \sqrt{N}$
  - E.g., modes of operation (counter-mode)

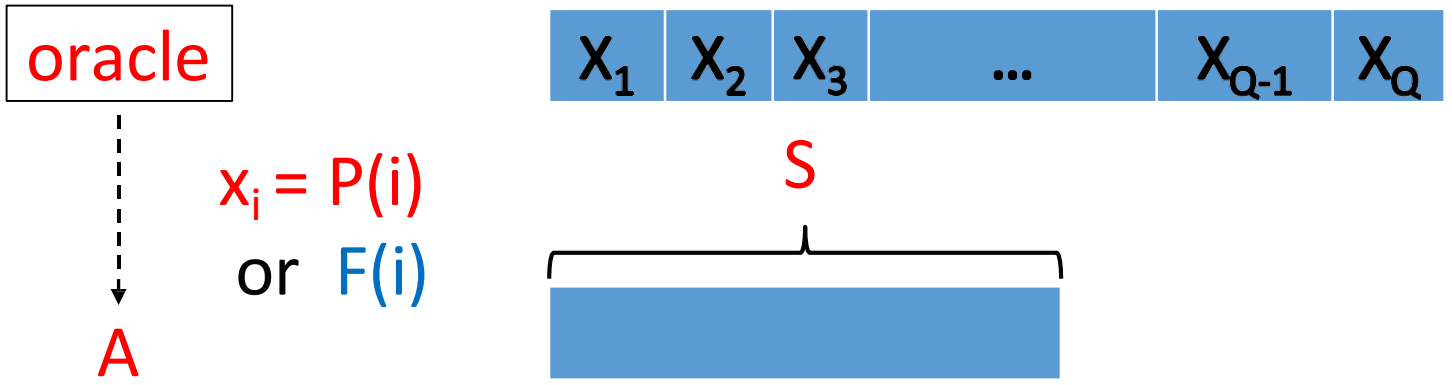


"Switching Lemma" for Random Permutation\Function

- "Switching Lemma": A has **advantage** bounded by $O(Q^2/N)$
  - $|\ \Pr[A^{P(\cdot)} = 1] - \Pr[A^{F(\cdot)} = 1]\ | \in O(Q^2/N)$
- Matching algorithm: **store** the Q query outputs and look for **collision** ($F(q_i) = F(q_j)$ for $q_i \neq q_j$)

Memory-Restricted Adversaries

- Algorithm requires **memory** ≈Q bits
- What about **memory-restricted** adversaries?
- Use **cycle detection algorithm** to obtain **optimal** $O(Q^2/N)$ advantage with ≈log(N) memory
- Requires **adaptive queries** to primitive
- What if adversary with S memory bits only given **stream** of Q elements produced by **random function\permutation**?
- Considered by Jaeger and Tessaro at EUROCRYPT 2019 [JT'19]



Streaming Switching Lemma [JT'19]

- "Streaming switching lemma" [JT'19]: adversary with S bits of memory with (1-pass) access to stream of Q elements from **random permutation\function** has distinguishing advantage of **at most** $\sqrt{Q \cdot S/N}$

- Application: better **security bounds** against **memory-restricted** adversaries for some modes of operation

Streaming Switching Lemma [JT'19]

- Application: better **security bounds** against **memory- restricted** adversaries for some modes of operation
- AES-based counter-mode:
- m_i encrypted to $(r_i , c_i = AES_K(r_i) \oplus m_i)$ for uniform r_i
- Eavesdropping adversary sees stream $(r_1 , c_1), (r_2 , c_2),...$

- **Replace** AES by random P + **apply** streaming switching lemma (several times):
- show $(r_1 , c_1), (r_2 , c_2),...$ Indistinguishable from
- $(r_i, \alpha_i) , (r_i , \alpha_i),...$ for uniform α_i

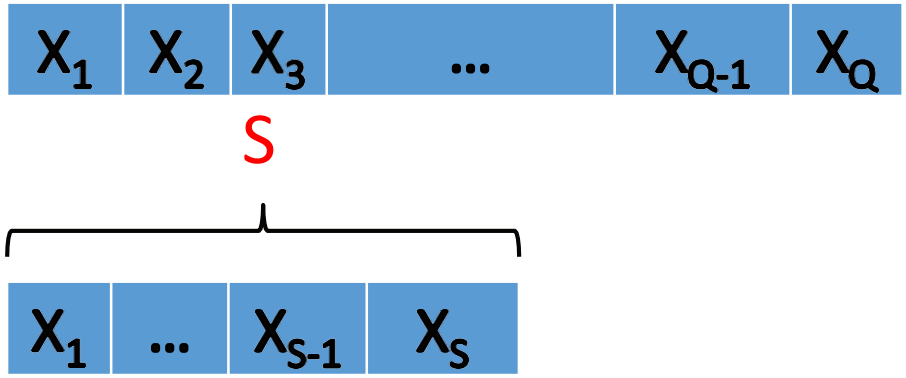
Streaming Switching Lemma

- "Streaming switching lemma" [JT'19]: adversary with S bits of memory with access to stream of Q elements from **random permutation\function** has distinguishing advantage of **at most** $\sqrt{Q \cdot S/N}$
- Application: if S is limited, counter-mode secure **beyond birthday bound**

- Limitations of [JS'19]:
- 1) Proof based on unproven combinatorial **conjecture**
- 2) Bound $\sqrt{Q \cdot S/N}$ **not tight** when $Q \cdot S \ll N$
  - E.g., when $S = Q$, bound is $\sqrt{Q^2/N}$, but (original) switching lemma gives Q^2/N

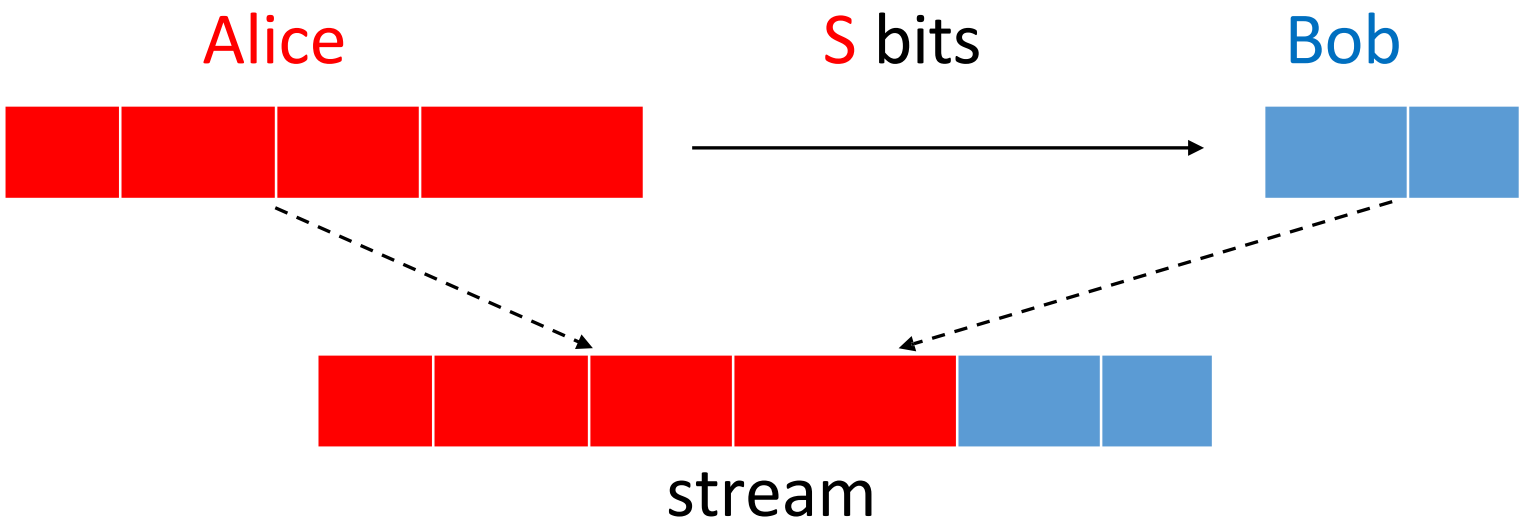
New Streaming Switching Lemma

- In this work: overcome limitations
- **New** streaming switching lemma bound $O(\log Q \cdot Q \cdot S/N)$
- **Tight** (up to poly-log factors):
  - Algorithm: store **first** S elements and look for collision with Q elements
  - Advantage: $\approx Q \cdot S/N$
- Note: when $S = Q$, we get (original) switching lemma



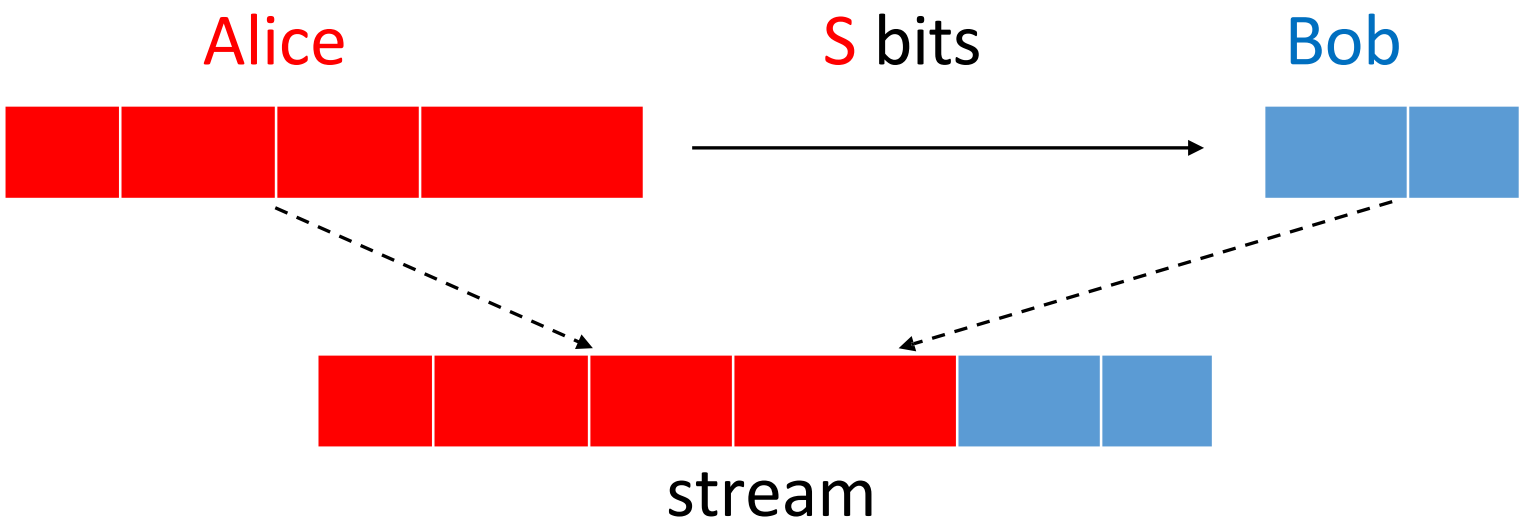
CC → Streaming

- Main idea: **reduce** from **communication complexity** (**CC**) problem (with **strong lower bounds**) to streaming
- General reduction framework from **one-way** CC problem:
  - Alice, Bob **solve CC problem** given access to **streaming** algorithm:
  - View concatenated inputs as stream
  - Alice **simulates** streaming algorithm on her input, **passes state** to Bob which continues simulation, outputs result



CC → Streaming

- Streaming algorithm with **memory** S gives **one-way** communication protocol with communication **cost** S (and **same advantage**)
- Lower bound on **cost** of communication protocol → lower bound on **memory** of streaming algorithm



Reduction Attempt for Random Permutation\Function

- Attempt: CC problem – each player gets Q/2 elements, chosen using ran permutation\function
- Useless: CC problem is **easy**
  - E.g., if $Q > \sqrt{N}$, players can **trivially distinguish** between permutation\function with **no communication**
  - Each player has **unlimited resources** and can detect a collision locally

Alice

$x_1,\ldots,x_{Q/2}$

Bob

$x_{Q/2+1},\ldots,x_Q$

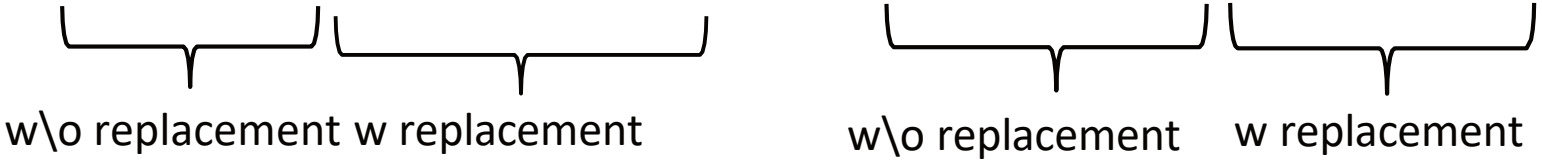
Reduction Attempt for Random Permutation\Function

- General restriction: in **hard CC problem** joint distributions for Alice and Bob's inputs should have **identical marginals**
  - Alice and Bob should have same **local view**
- **Impossible** when considering rand permutation\function distributions

- Solution: use **hybrid argument**
  - Consider **intermediate hybrid distributions** between random permutation and random function
  - Prove indistinguishability of **neighboring hybrid distributions** by reduction from CC

Hybrid Argument

- Attempt: define Q hybrids games
  - Game i: $x_1, \ldots \boldsymbol{x_{Q-i}}, x_{Q-i+1}, \ldots, x_Q$ or $x_1, \ldots x_{Q-i-1}, \boldsymbol{x_{Q-i}}, \ldots, x_Q$



- (Standard) hybrid argument **far from tight**
  - (Distinguishing advantage) x (num of hybrids) **too large**

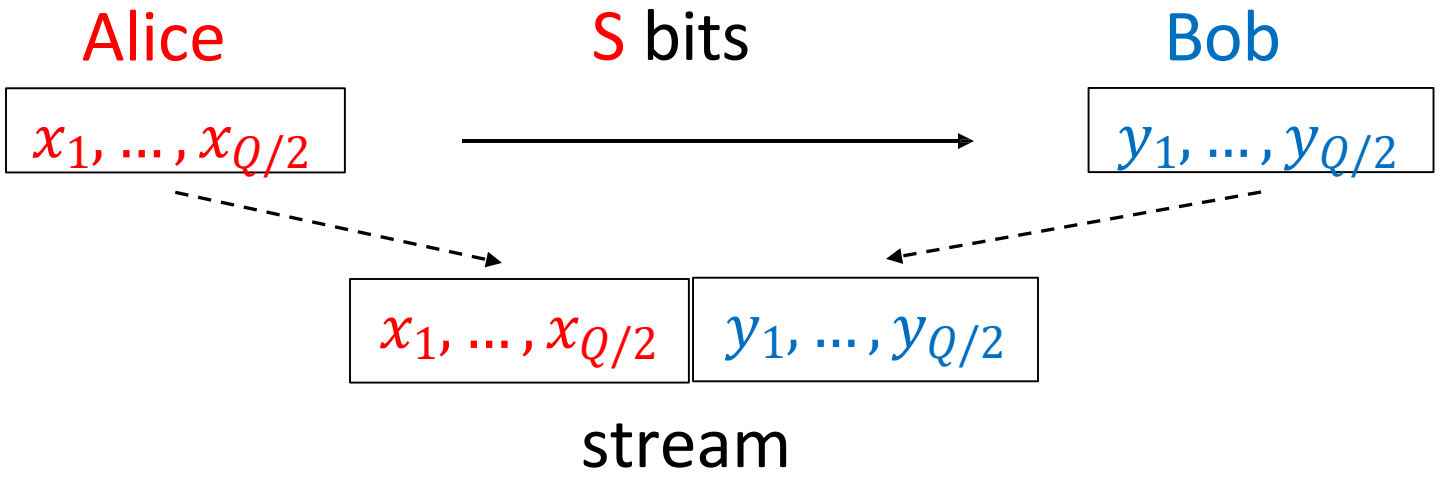
Improved Hybrid Argument

- Main idea: break dependency between **halves**
- Denote 1st sequence by $x_1, x_2, \ldots, x_{Q/2}, y_1, y_2, \ldots, y_{Q/2}$
- 1st distribution: elements chosen using (**same**) **permutation**
- 1st **intermediate** hybrid: $x_1, x_2, \ldots, x_{Q/2}$ and $y_1, y_2, \ldots, y_{Q/2}$ chosen using **independent permutations**

- Reduction from (one-way) CC:
- Alice gets 1st half of sequence, Bob gets 2nd half (decide if they obtain **same** or **independent** permutations)
  - Marginals are identical

Permutation Dependence

- (one way) CC problem - **permutation dependence** (**PDEP**):
- Alice and Bob decide if their inputs were drawn using **same** or **independent** permutations
- **PDEP** to **streaming** reduction:



UDISJ-> PDEP

- Communication **cost \ advantage** tradeoff for **PDEP**?
- Reduction from (unique) **disjointness (UDISJ)**
  - Each player receives a set of size n (domain size $O(n)$), need to decide if sets **intersect** or **disjoint**
- Theorem (informal)[BM'13, GW'14]**:** if Alice and Bob communicate c bits for **DISJ** (**UDISJ**) in the **worst case**, their **max advantage** is $O(c/n)$
  - Even when given access to **public randomness**

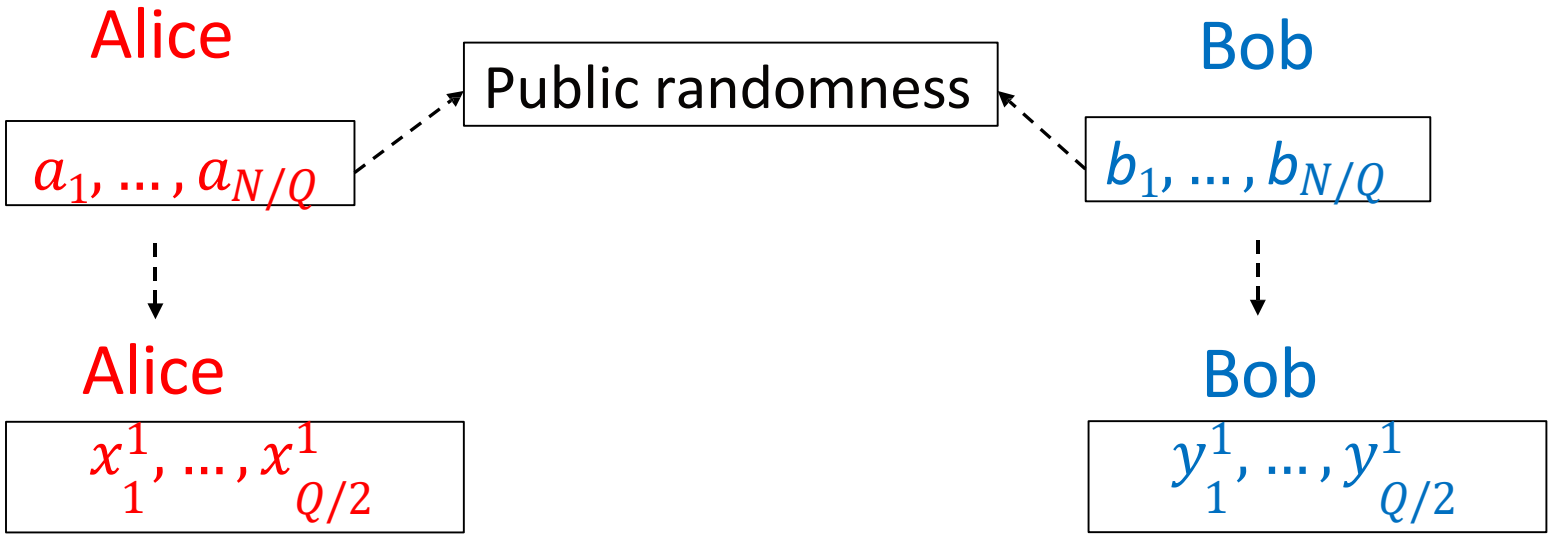
Alice

$a_1, \dots, a_n$

Bob

$b_1, \dots, b_n$

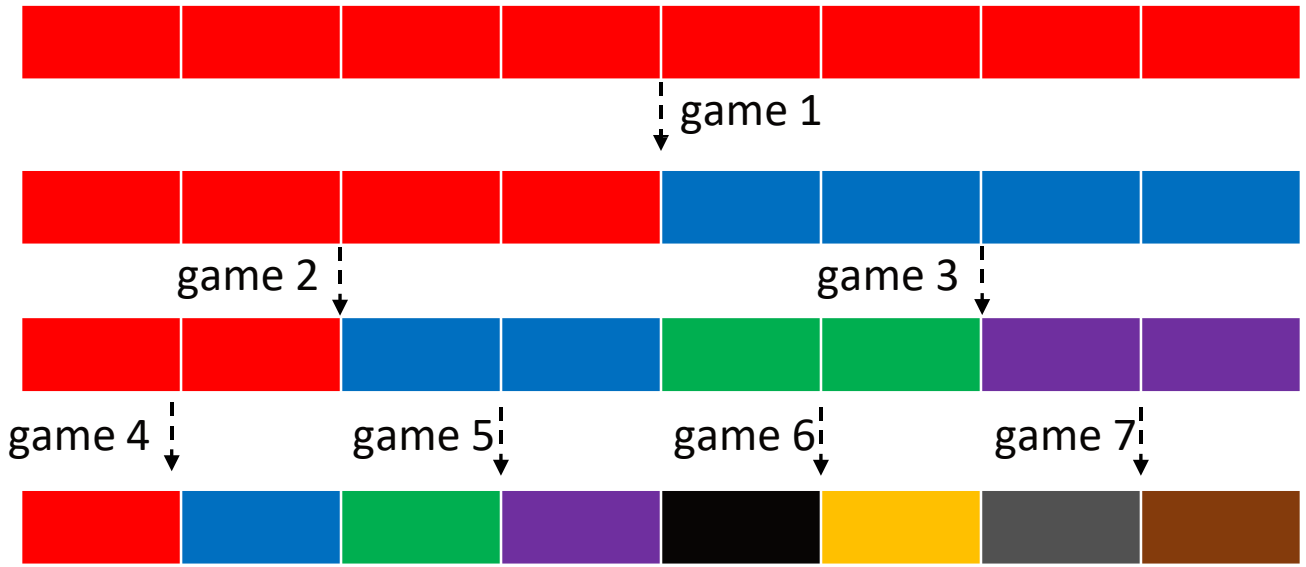
UDISJ-> PDEP



- Theorem (informal): there is a public coin **local** reduction that converts a **UDISJ** instance of size n=N/Q to a **PDEP** instance of size Q
  - **Shorter** inputs **harder** from **PDEP**, but **easier** for **UDISJ**
- Overall: UDISJ -> PDEP-> streaming bounds **max advantage** for hybrid game by $O(\mathrm{c}/\mathrm{n}) = O(S/(N/Q)) = O(Q \cdot S/N)$

The Full Hybrid Argument

- Once dependency between 2 halves broken:
  - Continue recursively (tree structure)
- 2'nd level: 2 games of distinguishing stream distributions on Q/2 elements
- Final distribution: Q elements divided into Q independent permutations == random function
- **Max advantage** for each level: $O(Q \cdot S/N)$
- **Total** max advantage: $O(\log Q \cdot Q \cdot S/N)$



Conclusions

- **New** streaming switching lemma bound $O(\log Q \cdot Q \cdot S/N)$
- **Tight** up to poly-log factors
- Reduction from CC to streaming uses **unconventional** hybrid argument
- Standard streaming problems defined in **worst case setting**
  - Gives **freedom to choose** hard distributions for CC problem
- In our (cryptographic) setting **streams distributions fixed**
  - Hybrid argument reduction applicable to more problems?
- Extension: **multi-pass** streaming switching lemma
  - Streaming alg allowed multiple passes over data

Thanks for your attention!