Extending Oblivious Transfers Efficiently

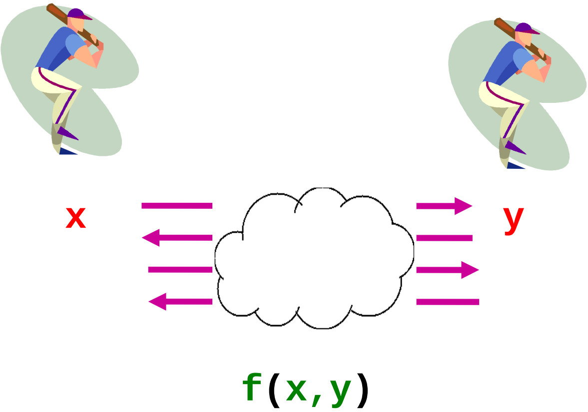
Yuval Ishai
Technion

Joe Kilian
NEC

Kobbi Nissim
Microsoft

Erez Petrank
Technion

Motivation



- How (in)efficient is *generic* secure computation?

myth

don' t even think about it

garbled circuit method

O(|x|) pub.

O(|f|) sym.

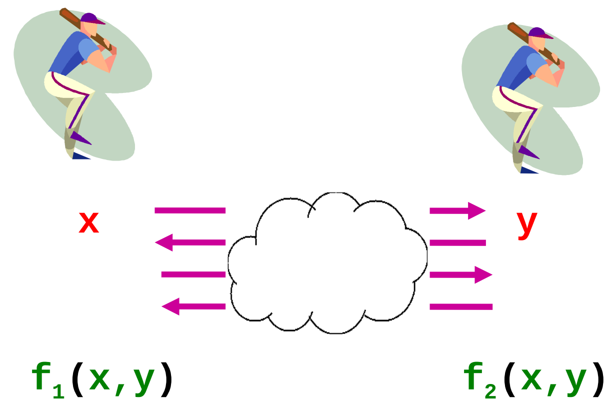
THIS WORK

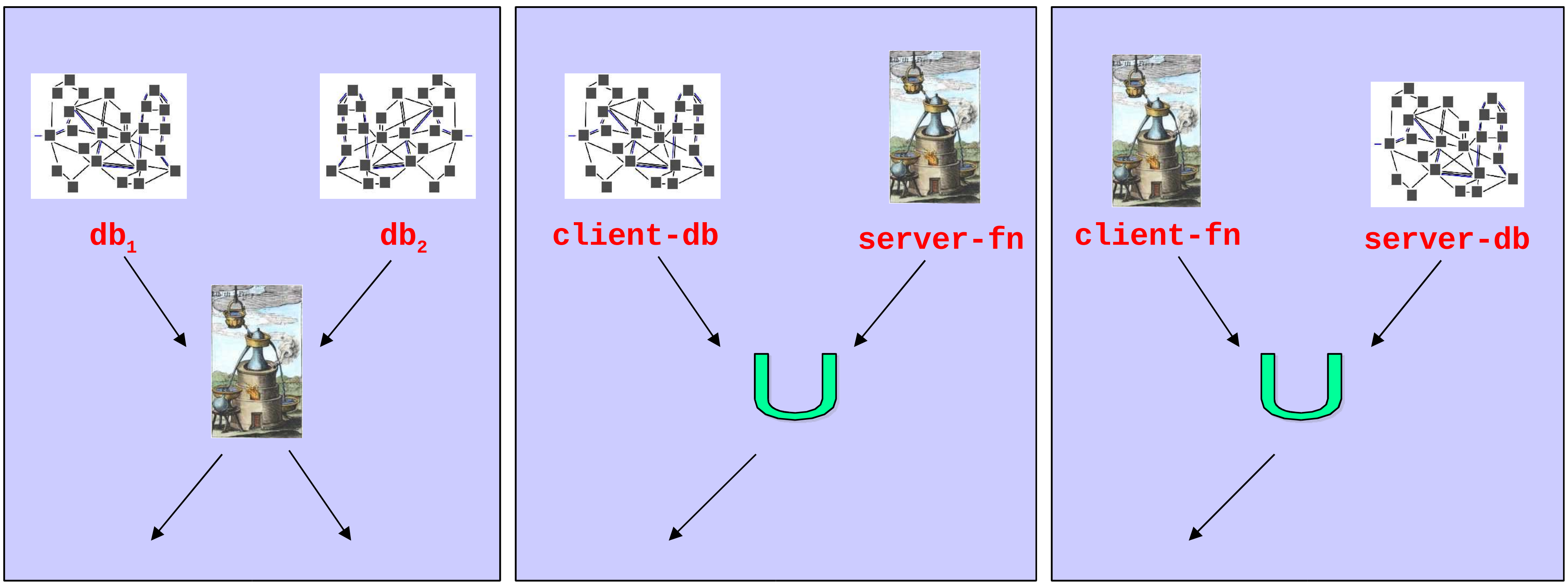
k pub.

O(|f|+|x|) sym.

sftp f.txt

Motivation



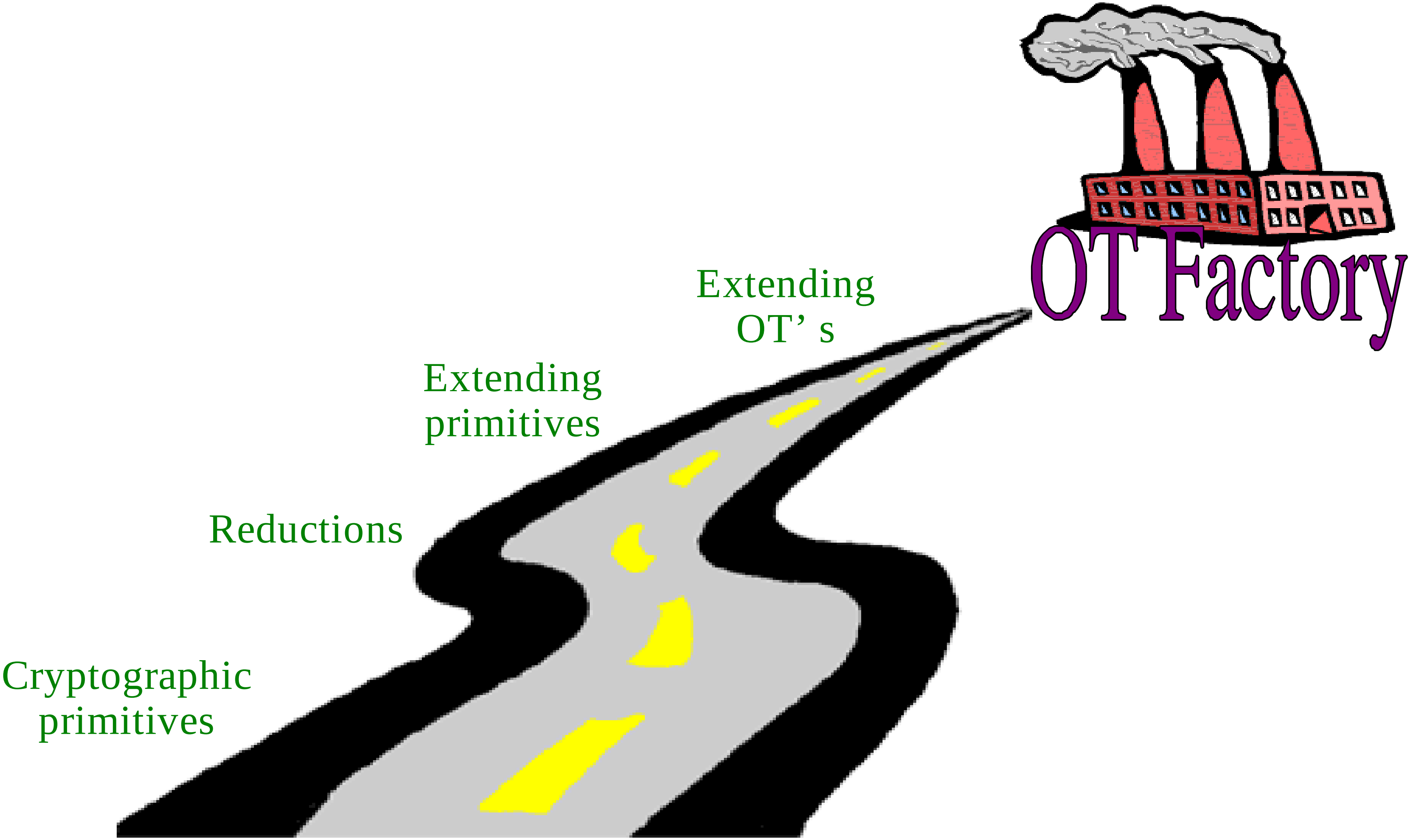


Efficiency of Secure Computation

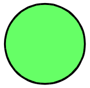
- Sometimes can use special structure of given functionality.
- Otherwise need to resort to *generic* techniques.
- How (in)efficient is generic secure computation?

myth	garbled circuit method	THIS WORK
	$O(\|x\|)$ pub.	k pub.
	$O(\|f\|)$ sym.	$O(\|f\|+\|x\|)$ sym.
don’t even think about it		`sftp f.txt`

Road Map


OT Factory
Extending
OT' s
Extending
primitives
Reductions
Cryptographic
primitives

A Taxonomy of Primitives

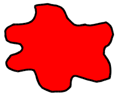


Symmetric encryption

Commitment

PRG

Collision resistant hashing



Public-key encryption

Key agreement

Oblivious transfer

Secure function evaluation

here you go

u

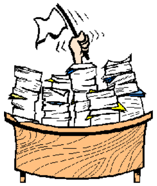
check this out

kidding?

nice try…

crack this!!!

hmmm…



here you go

u

check this out

kidding?

out

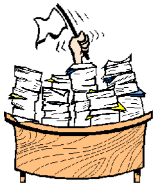
kidding?

crack this!!!

r u

kidding?

⋮



Symmetric encryption

Commitment

PRG

Collision resistant hashing

easy to implement heuristically (numerous candidates, may rely on “structureless” functions)

very cheap in practice

Public-key encryption

Key agreement

Oblivious transfer

Secure function evaluation

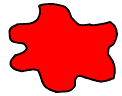
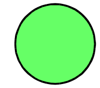
hard to implement heuristically (few candidates, rely on specific algebraic structures)

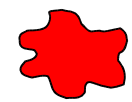
more expensive by orders of magnitude

Major challenge: bridge efficiency gap

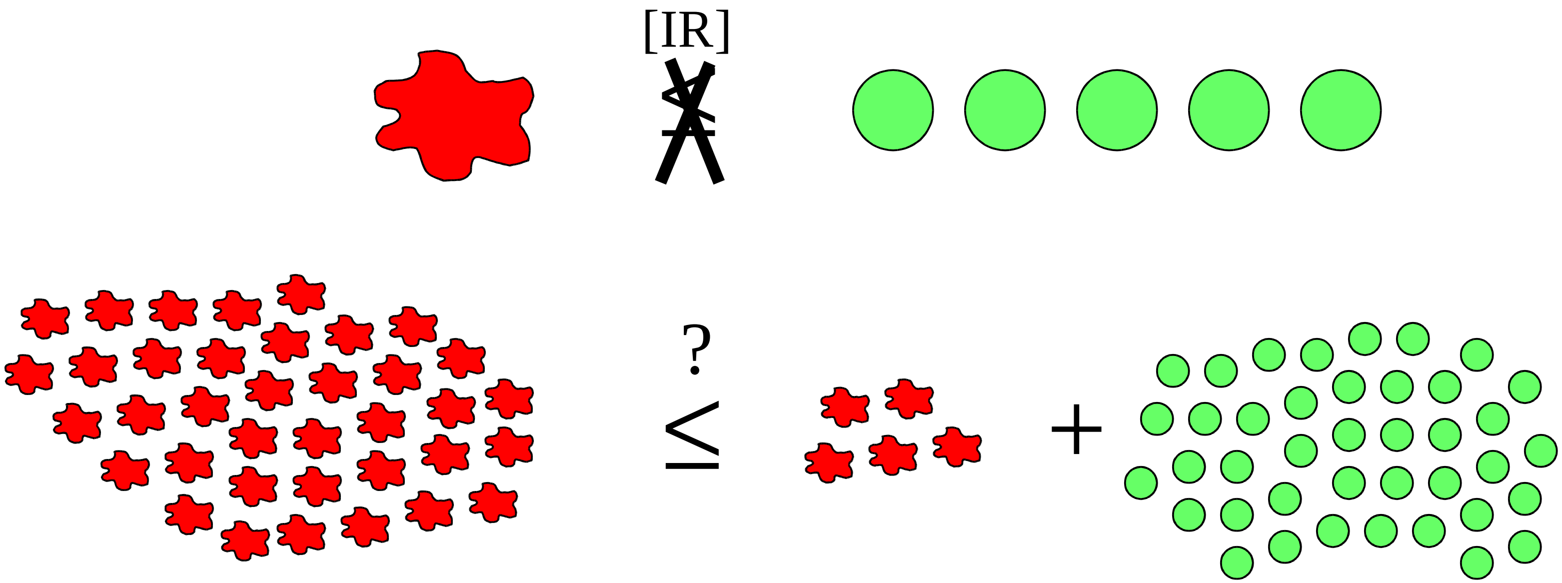
Reductions in Cryptography

- Motivated by
  - minimizing assumptions
  - gaining efficiency
- Reduction from Y to X: a mapping f such that if A implements X then $f(A)$ implements Y.
  - Cannot be ruled out when Y is believed to exist.
- Black-box reduction:
  - $f(A)$ makes a black-box use of A;
  - Black-box proof of security: Adversary breaking $f(A)$ can be used as a black box to break A.
- Almost all known reductions are black-box.
  - Non-black-box reductions are inefficient in practice.

Can  be reduced to ?

- Impagliazzo-Rudich [IR89]:
  No *black-box* reduction exists.
  - In fact, even a random oracle unlikely to yield 

Extending Primitives



Extending Y using X:

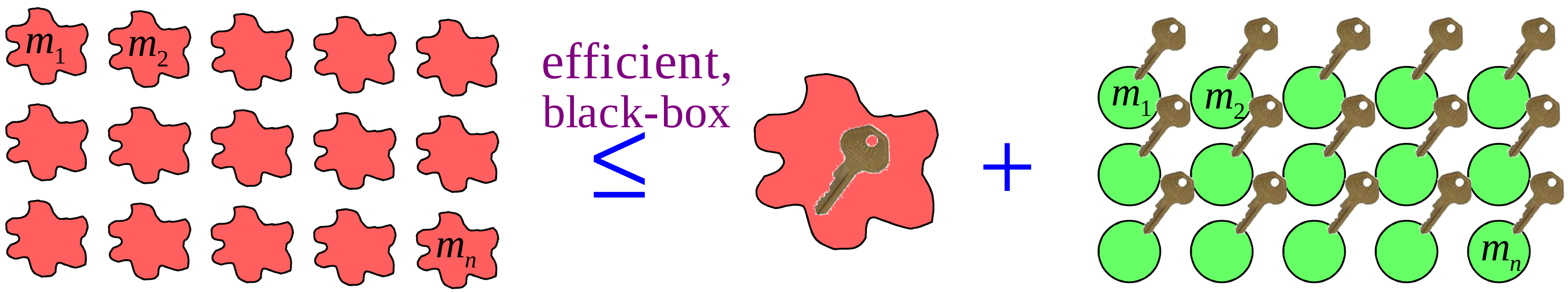
Realizing *n* instances of Y by making

- *k* (black-box) calls to Y, *k*<*n*
- arbitrary use of X

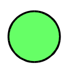
Want:

- *k* << *n*
- *black-box* use of X.

The Case of Encryption



- Extending PKE is easy…
- Huge impact on our everyday use of encryption.

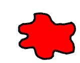


Symmetric encryption

Commitment

PRG

Collision resistant hashing



Public-key encryption

Key agreement

Oblivious transfer

Secure function evaluation

This work: Establish a similar result for remaining tasks.

Oblivious Transfer (OT)

- Several equivalent flavors [Rab81,EGL86,BCR87]
- $\binom{2}{1}$-OT:

Receiver | Sender

$r \in \{0,1\}$ | $x_0, x_1 \in \{0,1\}^l$

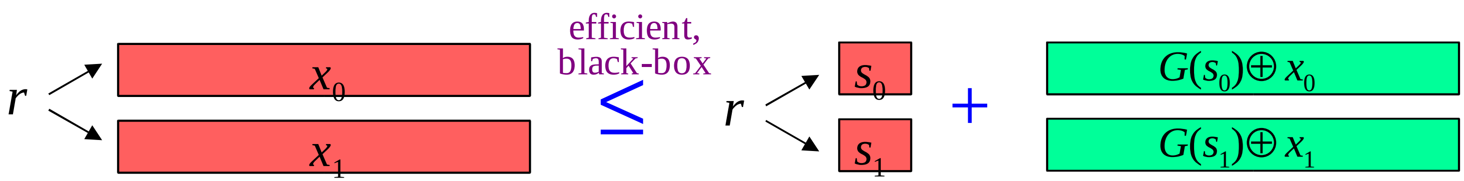
x_r | ???

- Formally defined as an instance of secure 2-party computation:
  – $\mathrm{OT}(r, \langle x_0, x_1 \rangle) = (x_r, \perp)$
- Extensively used in
  – general secure computation protocols [Yao86,GV87,Kil88,GMW88]
    - Yao’ s protocol: # of OT’ s = # of input bits
  – special-purpose protocols
    - Auctions [NPS99], shared RSA [BF97,Gil99], information retrieval [NP99], data mining [LP00,CIKRRW01],…

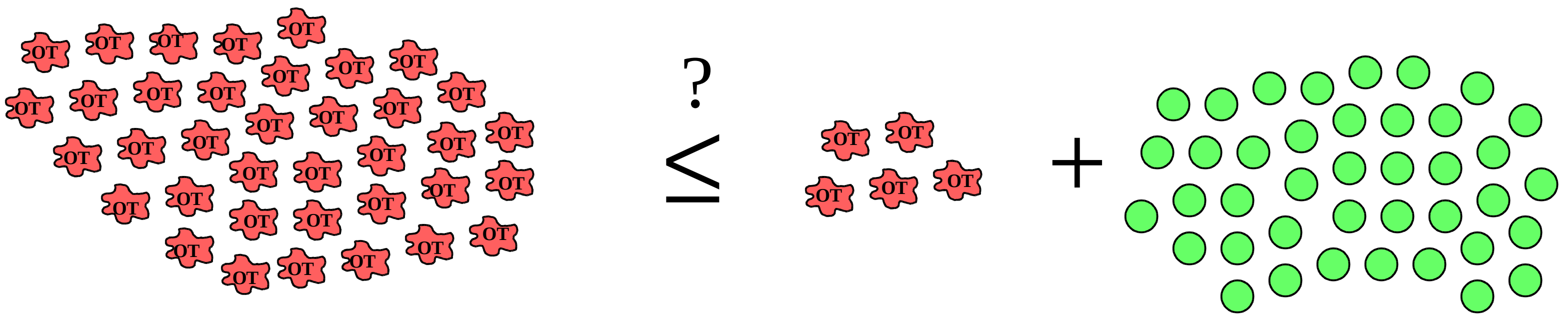
Cost of OT

- OT is at least as expensive as key-agreement.
  - OT' s form the efficiency bottleneck in many protocols.
  - "OT count" has become a common efficiency measure.
  - Some amortization was obtained in [NP01].

- Cost of OT is pretty much insensitive to l
  - Most direct OT implementations give l = security parameter "for free"
  - Handle larger l via use of a PRG

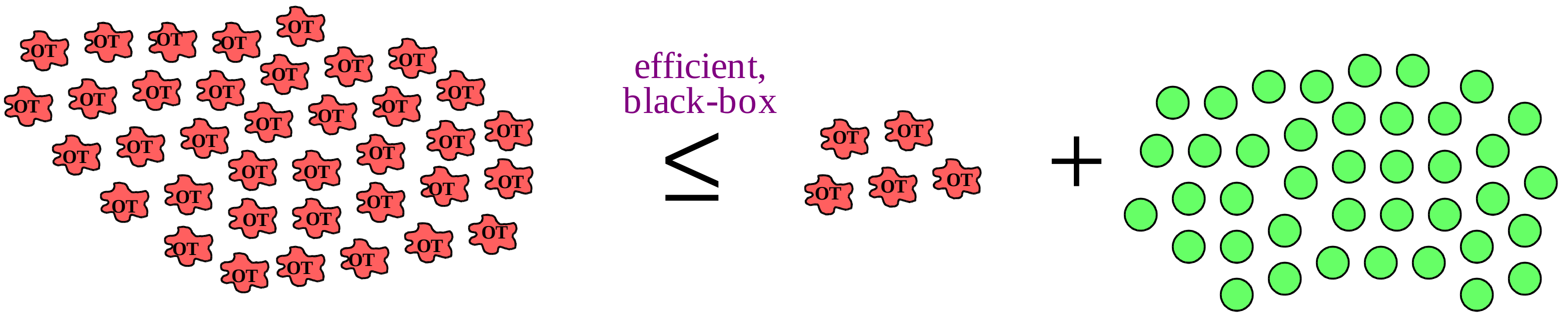


Extending Oblivious Transfers



- Beaver ' 96: OT can be extended using a PRG!!
  - Thm. If PRG exists, then k OT' s can be extended to $n=k^c$ OT' s.
- However:
  - Extension makes a non-black-box use of underlying PRG.
  - Numerous PRG invocations
  - Huge communication complexity
  - Unlikely to be better than direct OT implementations
- Can OT be extended via a black-box reduction?

Our Result

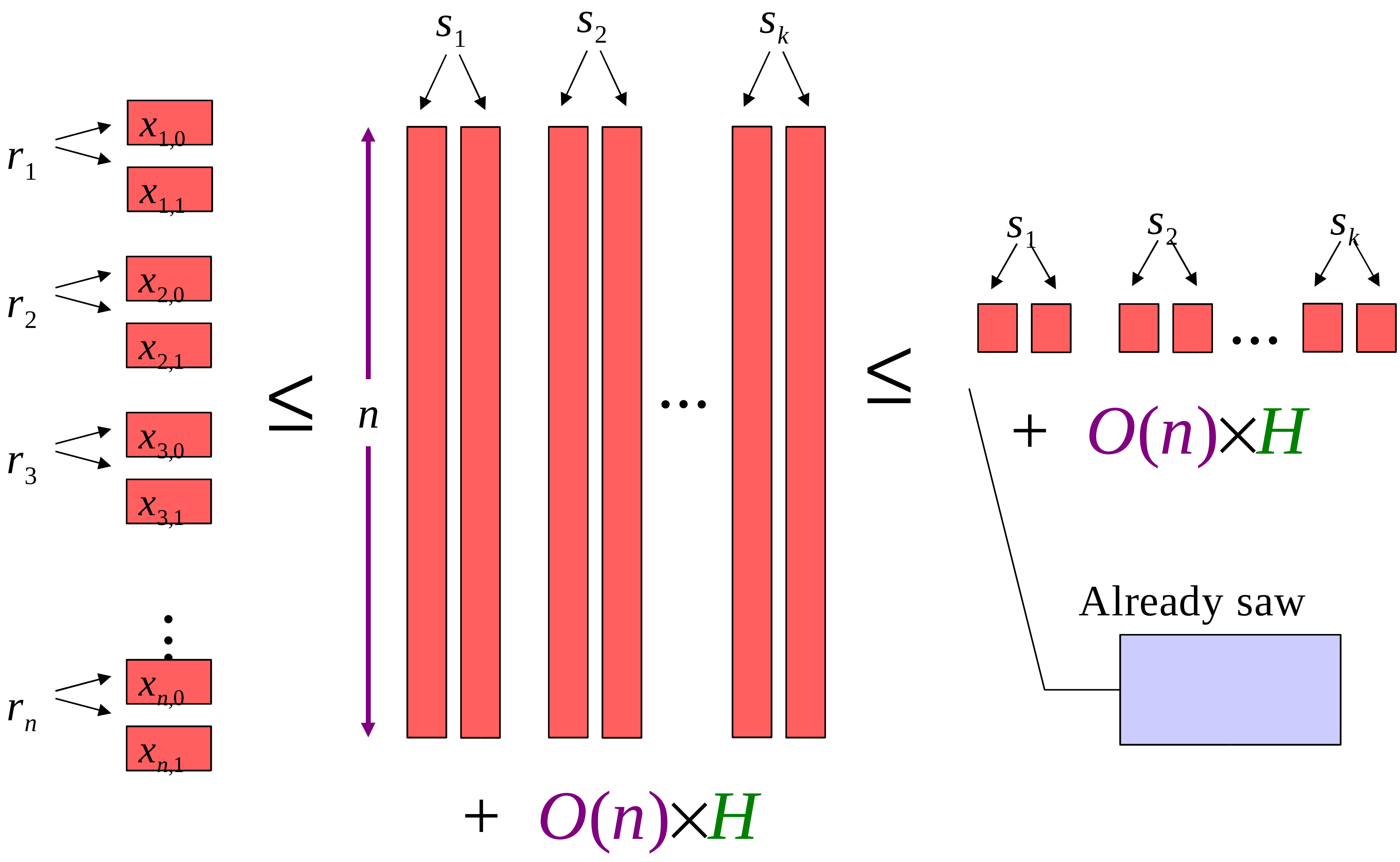


= random oracle

or

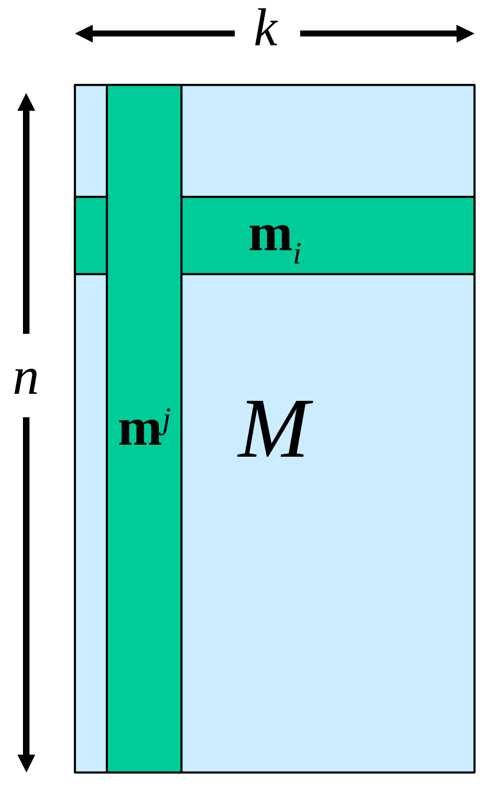
= new type of hash function

Strategy



r_1
$x_{1,0}$
$x_{1,1}$
r_2
$x_{2,0}$
$x_{2,1}$
r_3
$x_{3,0}$
$x_{3,1}$
r_n
$x_{n,0}$
$x_{n,1}$
$\leq$
n
s_1
s_2
s_k
$\leq$
s_1
s_2
s_k
$+ \; O(n)\times H$
Already saw
$+ \; O(n)\times H$

Notation

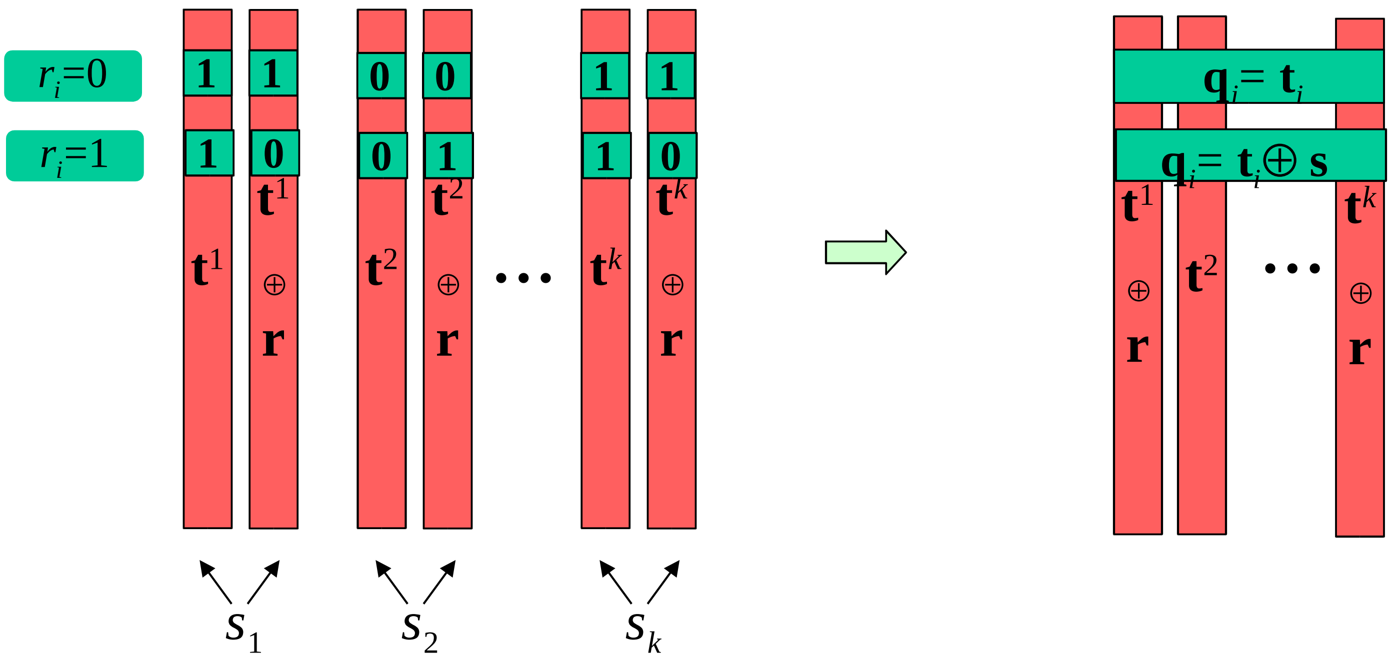


The Basic Protocol

Receiver picks $T \in_R \{0,1\}^{n\times k}$

Sender picks $\mathbf{s} \in_R \{0,1\}^{k}$

Sender obtains $Q \in \{0,1\}^{n\times k}$



- For $1 \le i \le n$, Sender sends $$y_{i,0} = x_{i,0} \oplus H(i, \mathbf{q}_i)$$ $$y_{i,1} = x_{i,1} \oplus H(i, \mathbf{q}_i \oplus \mathbf{s})$$
- For $1 \le i \le n$, Receiver outputs $z_i = y_{i,r_i} \oplus H(i, \mathbf{t}_i)$

Security

Receiver picks $T \in_R \{0,1\}^{n \times k}$

Sender picks $\mathbf{s} \in_R \{0,1\}^k$

Sender obtains $Q \in \{0,1\}^{n \times k}$

$r_i=0$ | $\mathbf{q}_i = \mathbf{t}_i$

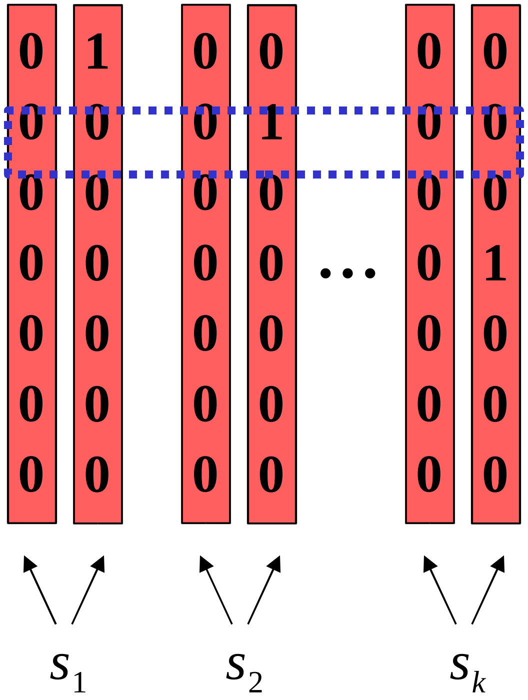
$r_i=1$ | $\mathbf{q}_i = \mathbf{t}_i \oplus \mathbf{s}$

Sender learns nothing
- Q is uniformly random

Receiver learns no additional info except w/neg prob.
- Must query H on $(i, \mathbf{t}_i \oplus \mathbf{s})$

- For $1 \le i \le n$, Sender sends $y_{i,0} = x_{i,0} \oplus H(i, \mathbf{q}_i)$, $y_{i,1} = x_{i,1} \oplus H(i, \mathbf{q}_i \oplus \mathbf{s})$
- For $1 \le i \le n$, Receiver outputs $z_i = y_{i,r_i} \oplus H(i, \mathbf{t}_i)$

Attack by a Malicious Receiver



- $\mathbf{q}_i = \begin{cases} \mathbf{0}, & s_i=0 \\ \mathbf{e}_i, & s_i=1 \end{cases}$
- Receiver can easily learn s_i given a-priori knowledge of $x_{i,0}$
  - Recover mask $H(i,\mathbf{q}_i) = y_{i,0} \oplus x_{i,0}$
  - Find s_i by querying H

Handling Malicious Receivers

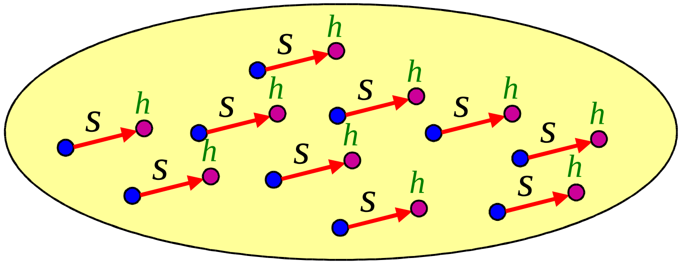
- Call Receiver *well-behaved* if each pair of rows are either identical or complementary.
- Security proof goes through as long as Receiver is well-behaved.
- Good behavior can be easily enforced via a cut-and-choose technique:
  - Run σ copies of the protocol using random inputs
  - Sender challenges Receiver to reveal the pairs it used in $\sigma/2$ of the executions. Aborts if inconsistency is found.
  - Remaining executions are combined.

Efficiency

- Basic protocol is extremely efficient
  - Seed of k OT's
  - Very few invocations of H per OT.
- Cut-and-choose procedure multiplies costs by $\approx \sigma$
  - Receiver gets away with cheating w/prob $\approx 2^{-\sigma/2}$
  - very small σ suffices if some penalty is associated with cheating
- Optimizations
  - Different cut-and-choose approach eliminates factor σ overhead to seed.
  - "Online" version, where the number n of OT's is not known in advance.

Eliminating the Random Oracle

- h:$\{0,1\}^k \rightarrow \{0,1\}^l$ is *correlation robust* if $f_s(t) := h(s \oplus t)$ is a weak PRF.
  - $(t_1, \ldots, t_n, h(s \oplus t_1), \ldots, h(s \oplus t_n))$ *is pseudorandom.*



- Correlation robust h can be used to instantiate H.
- Is this a reasonable primitive?
  - simple definition
  - satisfied by a random function
  - many efficient candidates (SHA1, MD5, AES, ..)

Conclusions

- OT’ s can be efficiently extended by making an efficient *black-box* use of a “symmetric” primitive.
  - Theoretical significance
    - Advances our understanding of relations between primitives
  - Practical significance
    - Amortized cost of OT can be made much lower than previously thought.
    - Significant even if OT did not exist: Initial seed of OT’ s can be implemented by physical means, or using multi-party computation.
    - Big potential impact on efficiency of secure computations

Further Research

- Assumptions
  - Can OT be extended using OWF as a black-box?
  - Study correlation robustness
- Efficiency
  - Improve efficiency in malicious case
- Scope
  - Obtain similar results for primitives which do not efficiently reduce to OT
- Practical implications
  - Has generic secure computation come to term?