Distributed Point Functions and their Applications

Niv Gilboa (BGU)
Yuval Ishai (Technion)

The concept

- Consider point functions
  - P_{xy}; $x \in \{0,1\}^n$, $y \in \{0,1\}^m$
- Point function
  - $P_{xy}(x')=0$ if $x' \neq x$ and $P_{xy}(x)=y$.
- Our goal is additively share a secret point function using a succinct representation
  - The shares are F_0 and F_1 s.t $P_{xy}=F_0 \oplus F_1$.

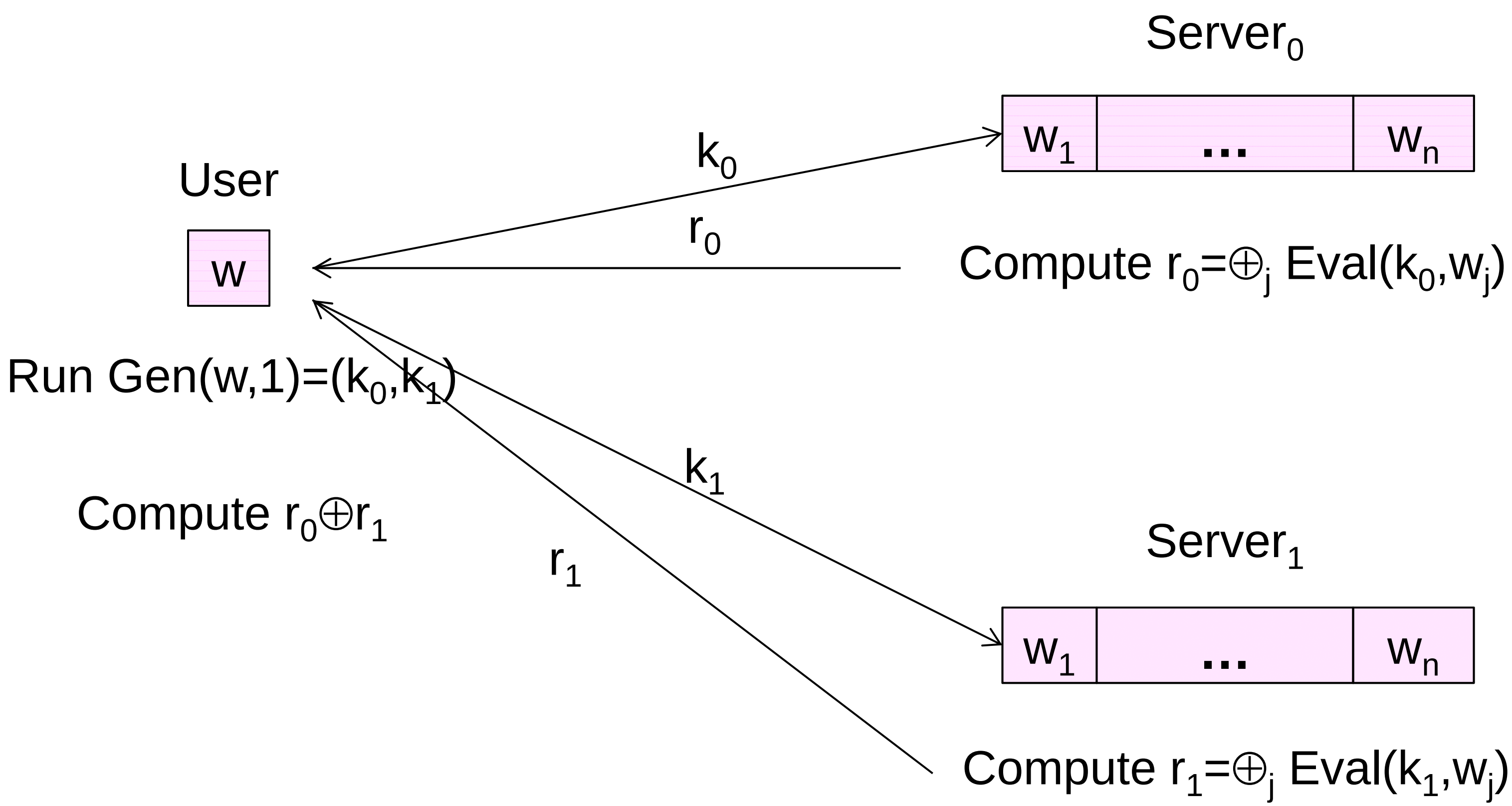
Model

- More formally, a Distributed Point Function (DPF) is two PPT algorithms
  - $Gen(x,y)=(k_0,k_1)$
  - $Eval(k,x')$
- Such that
  - Correctness - $P_{xy}(x')=Eval(k_0,x')\oplus Eval(k_1,x')$
  - Secrecy – k_0 (or separately k_1) can be simulated given only $|x|$ and $|y|$.

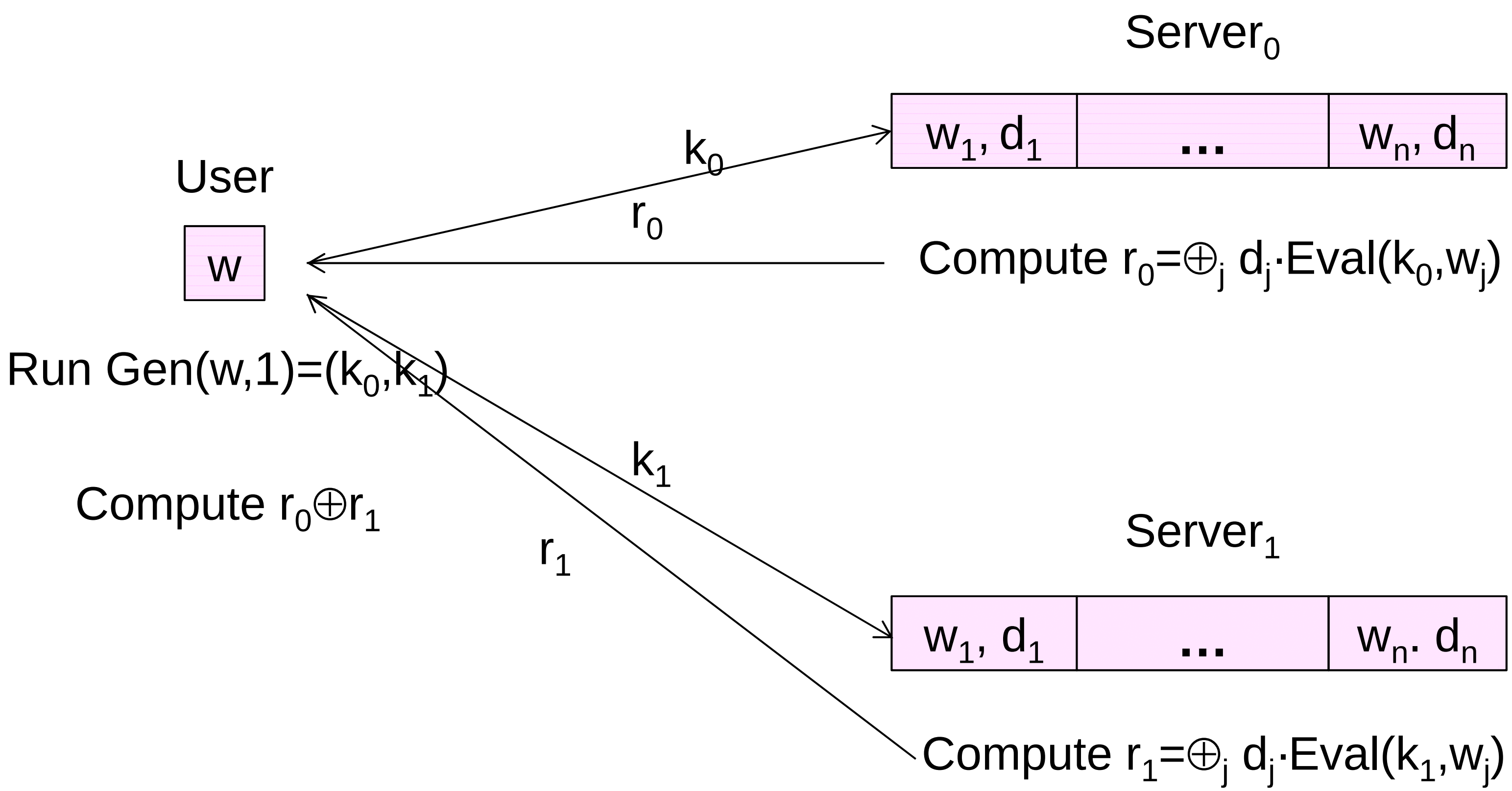
Motivation

- Why be interested in this question?
- Interesting applications!
- Two server Private Information Retrieval (PIR) [CGKS95].
- Two server Private retrieval by Keywords [CGN97, FIPR05, OS07].
- Private Information Storage [OS97].
- Worst-case to average case reductions [BF90, BFNW91].

Using DPF for keywords



Using DPF for keywords



Results

- Main theorem: OWF → DPF.
- Key size is short
  - For security parameter k (e.g. length of AES key) it is ~$8k \cdot |x|^{\log 3} + |y|$ bits.
- Converse: DPF → OWF

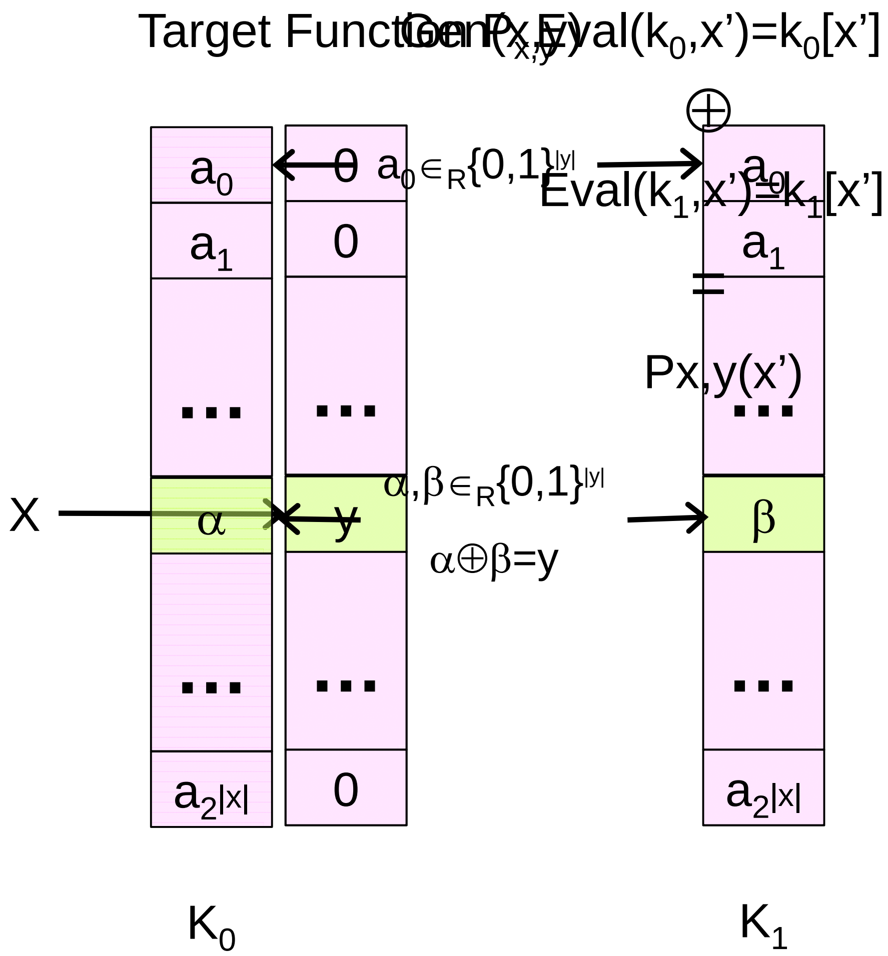
Exact numbers

|X|	Key size (in bytes)
20	300 bytes
40	~630 bytes
80	~2.25 Kbytes
160	~8 Kbytes

Results - applications

- PIR scheme -
  - First poly-logarithmic, constant server PIR scheme based on OWF.
  - First poly-logarithmic, binary, two server scheme (improving on [CG97]).
- PIR writing (storage) - similar to PIR results.
- Keyword search - first 2-server solution with 1-bit answers.
- Efficient worst-case to average-case 2-query reductions for PSPACE and EXPTIME complete languages.

Trivial Solution ($2^{|x|}$)



Improvement-Preliminaries

- Regard $P_{x,y}$ as two-dimensional.
- Instead of $P_{x,y}:\{0,1\}^{|x|}\rightarrow\{0,1\}^{|y|}$ think of

$$P_{(i,j),y}:\{0,1\}^{|x|/2}\times\{0,1\}^{|x|/2}\rightarrow\{0,1\}^{|y|}$$

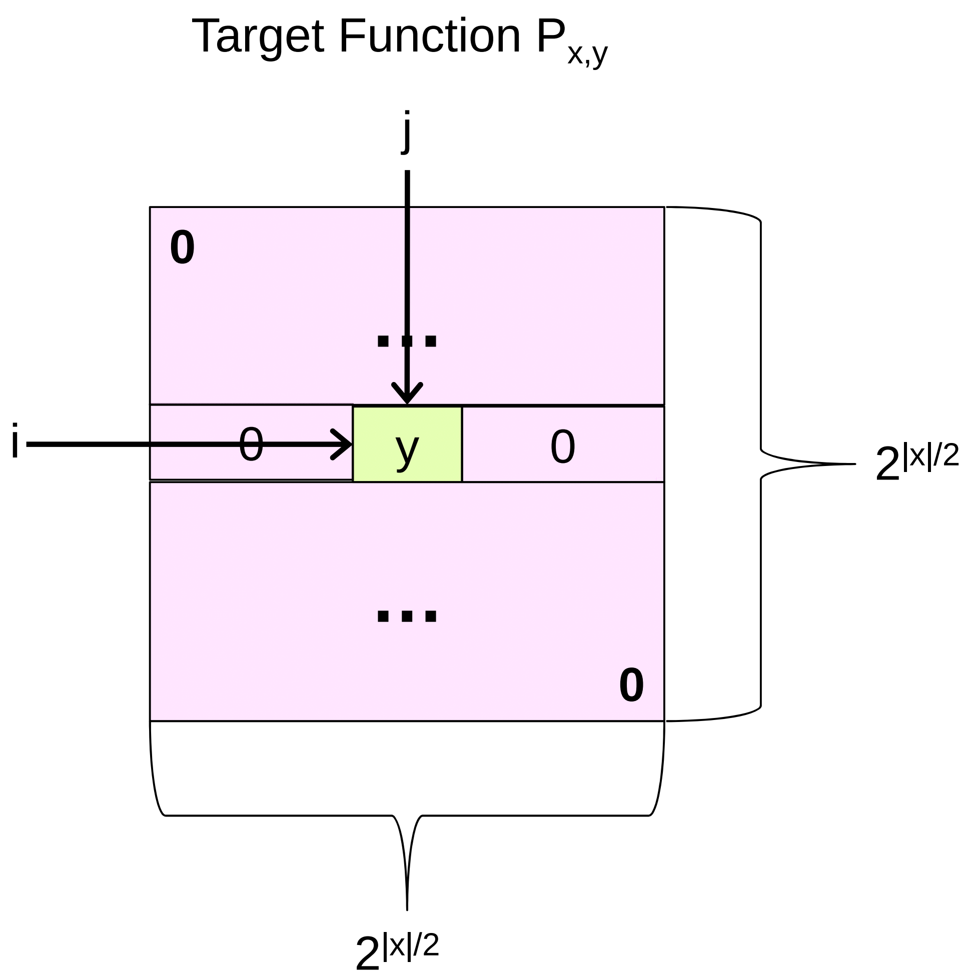
- Let G be a pseudo-random generator.

Preliminaries (cont.)

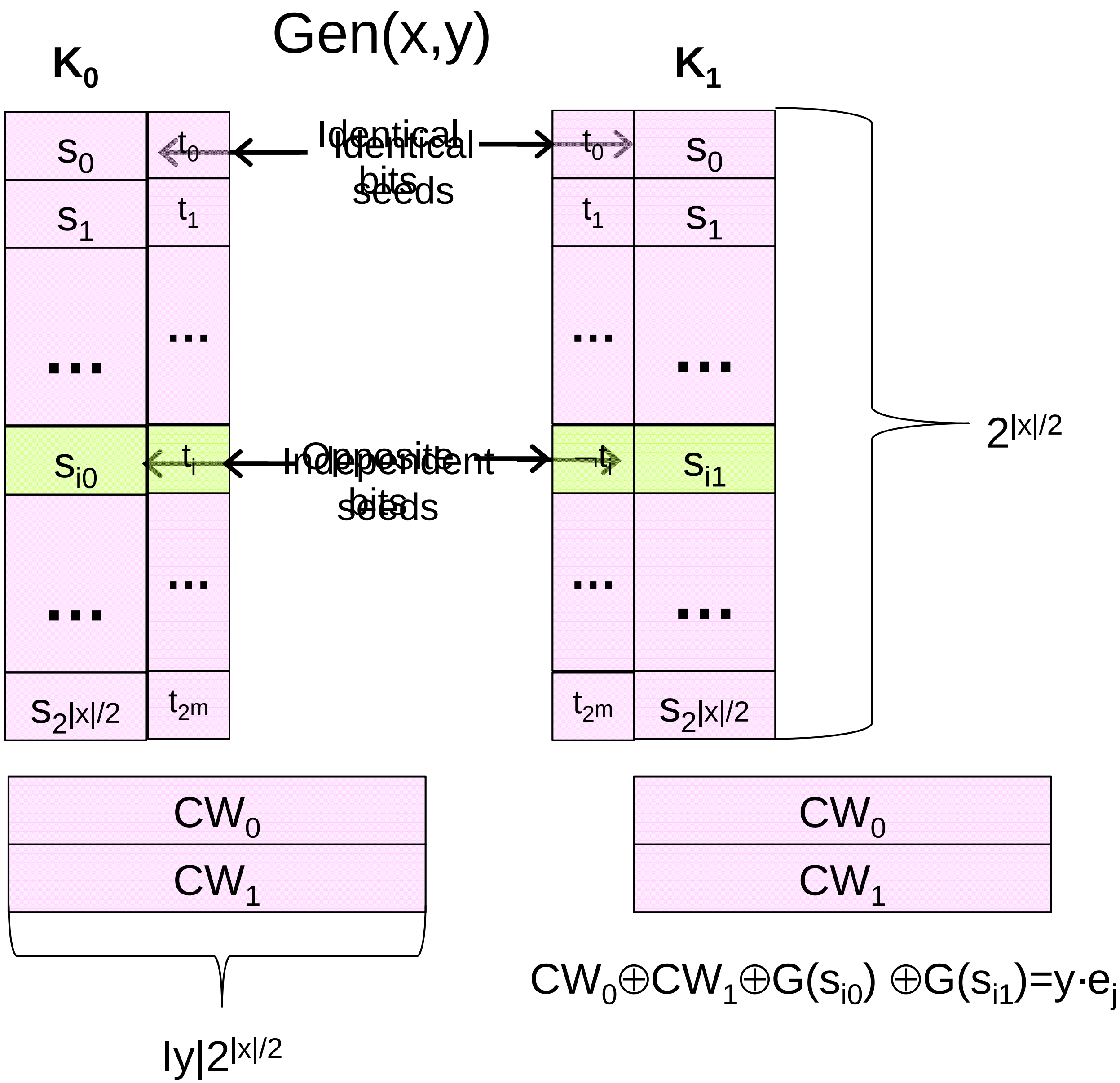
- What if Gen(x,y) produces
  - $k_0=(s_1,\ldots,s_{i0},\ldots,s_{2^{|x|/2}})$
  - $k_1=(s_1,\ldots,s_{i1},\ldots,s_{2^{|x|/2}})$
- For x' represented by (i',j'), let $Eval(k_0,x')=G(s_{i'})[j']$.
- Then

$$Eval(k_0,x') \oplus Eval(k_1,x') = \begin{cases} P_{xy}(x') & \text{if } i'=i \\ \text{Junk} & \text{if } i' \neq i \end{cases}$$

$2^{|x|/2}$ solution



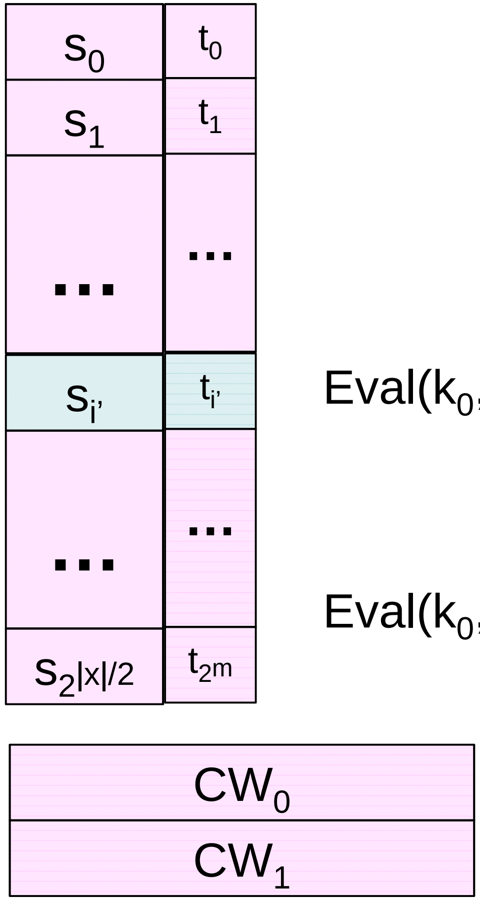
$2^{|x|/2}$ solution



$2^{|x|/2}$ solution

$Eval(k_0,x)$

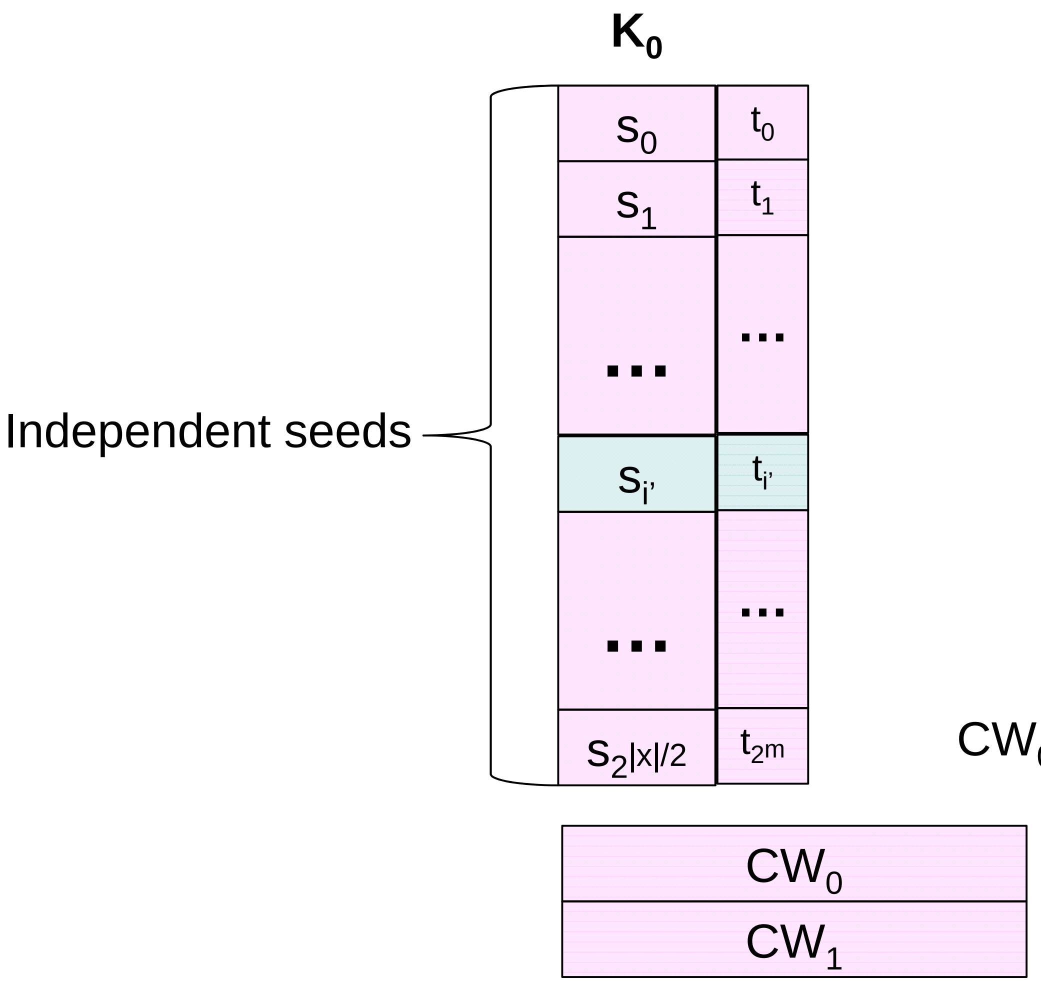
K_0 **$x'=(i',j')$**



$$Eval(k_0,x')=(G(s_{i'})\oplus CW_{t_{i'}})[j']$$

$$Eval(k_0,x')\oplus Eval(k_1,x')=\begin{cases}0 & (i',j')\neq(i,j)\\ y & (i',j')=(i,j)\end{cases}$$

Secrecy



$$CW_0 \oplus CW_1 \oplus G(s_{i0}) \oplus G(s_{i1}) = y \cdot e_j$$

Is this good or bad?

- We have a solution for a distributed point function.
- Oh! That's good!
- But the key length and running times are exponential in $|x|$ ($2^{|x|/2}|y|$ to be exact).
- Oh! That's bad!
- Can we improve the length and time?

Recursion - Gen

- Let's look at the Gen algorithm again.
- The keys k_0, k_1 are made up of
  - $2^{|x|/2}$ seeds - all identical except one
  - $2^{|x|/2}$ bits (t_i) - all identical except one
  - Two identical correction words CW0, CW1
- Call Gen(i, seed) recursively on domain of size $2^{|x|/2}$ seeds (plus bits).

Recursion - Gen (cont.)

- What about the two correction words?
- Recall: $CW_0 \oplus CW_1 = G(s_{i0}) \oplus G(s_{i1}) \oplus y \cdot e_j$
  - Exchange $y \cdot e_j$ by a call to *Gen*(j,y)
- Result – each step of recursion returns a key of length ≈3(previous length)$^{1/2}$.
- Stop recursion at shortest key length.

Recursion - Eval

- On a call Eval(k_0,x′) for x′=(i′,j′)
  - Parse k_0 as σ, CW_0, CW_1, where σ is the result of Gen on the seeds.
  - Run Eval recursively on σ to derive $s_{i'}$, $t_{i'}$
  - Compute $v=G(s_{i'})\oplus CW_{t_{i'}}$
  - Run Eval recursively on (v,j′) to obtain output.

Thank You!!!