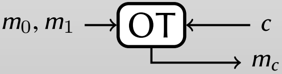
Oblivious Transfer

Mike Rosulek
Oregon State University OSU

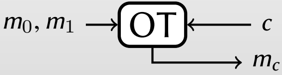
crypt@b-it 2018



OT recap

OT is . . .

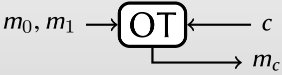
- Necessary for MPC [Kilian]
- **Inherently expensive**: impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]



OT recap

OT is . . .

- Necessary for MPC [Kilian]
- **Inherently expensive**: impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]

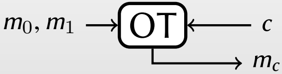


Today's agenda: reducing the cost of OT

1. **Precomputation:** can compute OTs even before you know your input!
2. **OT extension:** 128 OTs suffice for everything.

Random OT

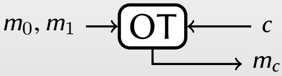
Standard OT:



Random OT:

Random OT

Standard OT:



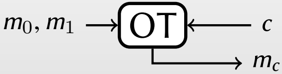
Deterministic functionality; parties choose all inputs

Random OT:

Randomized functionality chooses m_0, m_1, c uniformly.

Random OT

Standard OT:



Deterministic functionality;
parties choose all inputs

Random OT:

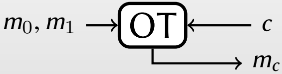
Randomized functionality
chooses m_0, m_1, c uniformly.

Beaver Derandomization Theorem [Beaver91]

There is a cheap protocol that securely evaluates an instance of **standard OT** using an instance of **random OT**.

Random OT

Standard OT:



Deterministic functionality;
parties choose all inputs

Random OT:

Randomized functionality
chooses m_0, m_1, c uniformly.

Beaver Derandomization Theorem [Beaver91]

There is a cheap protocol that securely evaluates an instance of **standard OT** using an instance of **random OT**.

Offline/online approach to 2PC:

- In **offline preprocessing phase**, generate many random OTs
- During **online phase**, OT inputs are determined — cheaply derandomize the offline OTs with Beaver's trick.

Beaver Derandomization [Beaver91]

$$m_0^{\$}, m_1^{\$} \longleftarrow \boxed{\text{OT}} \longrightarrow c^{\$}, m_{c^{\$}}^{\$}$$

offline

Beaver Derandomization [Beaver91]

$m_0^\$, m_1^\$ \longleftarrow$ OT $\longrightarrow c^\$, m_{c^\$}^\$$

offline

online

m_0, m_1 c m_c ?

Beaver Derandomization [Beaver91]

$$m_0^\$, m_1^\$ \longleftarrow \boxed{\text{OT}} \longrightarrow c^\$, m_{c^\$}^\$$$

offline

online

m_0, m_1 $\qquad$ c

$$x_0 = m_0^\$ \oplus m_0$$
$$x_1 = m_1^\$ \oplus m_1$$
$$\longrightarrow$$

- **Idea:** Alice can use $m_0^\$$ and $m_1^\$$ as one-time pads to mask m_0, m_1

Beaver Derandomization [Beaver91]

$m_0^\$, m_1^\$ \longleftarrow$ OT $\longrightarrow c^\$, m_{c^\$}^\$$

offline

- - - - - - - - - - - - - - - - - - - -

online

m_0, m_1 $\qquad$ $c \quad (= c^\$)$

$$x_0 = m_0^\$ \oplus m_0$$
$$x_1 = m_1^\$ \oplus m_1$$

$\longrightarrow$ compute $x_c \oplus m_{c^\$}^\$$

$$= x_c \oplus m_c^\$ = m_c$$

- **Idea:** Alice can use $m_0^\$$ and $m_1^\$$ as one-time pads to mask m_0, m_1
- If $c = c^\$$ this works: Bob can decrypt **only** m_c (no info about m_{1-c})

Beaver Derandomization [Beaver91]

$m_0^\$, m_1^\$ \longleftarrow$ **OT** $\longrightarrow c^\$, m_{c^\$}^\$$

offline

online

m_0, m_1 $\qquad\qquad$ $c \quad (\neq c^\$)$

$$x_0 = m_1^\$ \oplus m_0$$
$$x_1 = m_0^\$ \oplus m_1$$

$\longrightarrow$ compute $x_c \oplus m_{c^\$}^\$$

$$= x_c \oplus m_{1 \oplus c}^\$ = m_c$$

- **Idea:** Alice can use $m_0^\$$ and $m_1^\$$ as one-time pads to mask m_0, m_1
- If $c = c^\$$ this works: Bob can decrypt **only** m_c (no info about m_{1-c})
- If $c \neq c^\$$ Bob learns wrong m unless Alice swaps $m_0^\$, m_1^\$$.

Beaver Derandomization [Beaver91]

$m_0^\$, m_1^\$ \longleftarrow$ **OT** $\longrightarrow c^\$, m_{c^\$}^\$$

offline

- -

online

m_0, m_1 $\qquad\qquad$ c

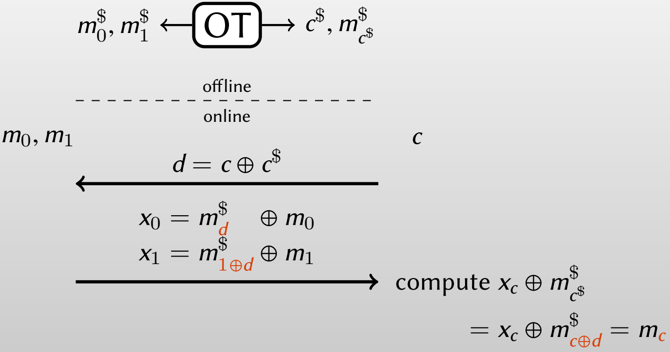
$d = c \oplus c^\$$ $\longleftarrow$

$$x_0 = m_{d}^\$ \oplus m_0$$
$$x_1 = m_{1 \oplus d}^\$ \oplus m_1$$

$\longrightarrow$

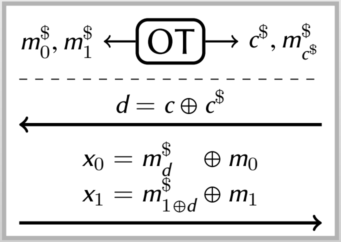
- **Idea:** Alice can use $m_0^\$$ and $m_1^\$$ as one-time pads to mask m_0, m_1
- If $c = c^\$$ this works: Bob can decrypt **only** m_c (no info about m_{1-c})
- If $c \neq c^\$$ Bob learns wrong m unless Alice swaps $m_0^\$, m_1^\$$.
- **Solution:** Bob says whether $c = c^\$$ (safe: Alice has no info about $c^\$$)

Beaver Derandomization [Beaver91]



- **Idea:** Alice can use $m_0^\$$ and $m_1^\$$ as one-time pads to mask m_0, m_1
- If $c = c^\$$ this works: Bob can decrypt **only** m_c (no info about m_{1-c})
- If $c \neq c^\$$ Bob learns wrong m unless Alice swaps $m_0^\$, m_1^\$$.
- **Solution:** Bob says whether $c = c^\$$ (safe: Alice has no info about $c^\$$)

Beaver Derandomization [Beaver91]



- **Offline cost:** same as before (1 OT instance)
- **Online cost:** simple XORs

E paucis plura

from a few, many

An analogy from encryption

Oblivious Transfer is inherently expensive:

- Impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]

An analogy from encryption

Oblivious Transfer is inherently expensive:

- Impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]

Public-key encryption is inherently expensive:

- Impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]

An analogy from encryption

Oblivious Transfer is inherently expensive:

- Impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]

Public-key encryption is inherently expensive:

- Impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]

PKE cost be **minimized** with hybrid encryption:

- Use (expensive) PKE to encrypt short s
- Use (cheap) symmetric-key encryption *with key s* to encrypt long M

PKE of λ bits + cheap SKE = PKE of N bits

An analogy from encryption

Oblivious Transfer is inherently expensive:

- Impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]

Is there an analog of "hybrid encryption" for OT?

λ instances of OT + cheap SKE = N instances of OT ??

Public-key encryption is inherently expensive:

- Impossible using only cheap crypto (random oracle) [ImpagliazzoRudich89]

PKE cost be **minimized** with hybrid encryption:

- Use (expensive) PKE to encrypt short s
- Use (cheap) symmetric-key encryption *with key s* to encrypt long M

PKE of λ bits + cheap SKE = PKE of N bits

Beaver OT extension [Beaver96]

Key insight: Yao's protocol requires only # of OTs proportional to function's **input length**

Beaver OT extension [Beaver96]

Key insight: Yao's protocol requires only # of OTs proportional to function's **input length**

Beaver protocol: Run the following 2PC using Yao:

$s_A \longrightarrow$ Use $s_A \oplus s_B$ as pseudorandom seed to: $\longleftarrow s_B \in \{0,1\}^\lambda$

- Sample $2n$ random strings $(m_{1,0}, m_{1,1}), \ldots, (m_{n,0}, m_{n,1})$.
- Sample n-bit random string r

$\{m_{i,b}\}_{i,b} \longleftarrow$ $\longrightarrow r, \{m_{i,r_i}\}_i$

Beaver OT extension [Beaver96]

Key insight: Yao's protocol requires only # of OTs proportional to function's **input length**

Beaver protocol: Run the following 2PC using Yao:

$s_A \longrightarrow$ Use $s_A \oplus s_B$ as pseudorandom seed to: $\longleftarrow s_B \in \{0,1\}^\lambda$

- Sample $2n$ random strings $(m_{1,0}, m_{1,1}), \ldots, (m_{n,0}, m_{n,1})$.
- Sample n-bit random string r

$\{m_{i,b}\}_{i,b} \longleftarrow$ (output) $\longrightarrow r, \{m_{i,r_i}\}_i$

- # OTs = input length = λ
- Output provides $n \gg \lambda$ instances of OT (random strings + choice bits)
- Impractical **feasibility** result (2PC evaluation of a PRG circuit)

Yuval Ishai, Joe Kilian, Kobbi Nissim, Erez Petrank:
Extending Oblivious Transfers Efficiently.
Crypto 2003.

IKNP protocol [IshaiKilianNissimPetrank03]

r
1
0
0
0
1
0
1
1
⋮

Bob

- Bob has input r

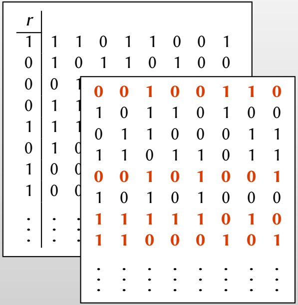
IKNP protocol [IshaiKilianNissimPetrank03]

r								
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Bob

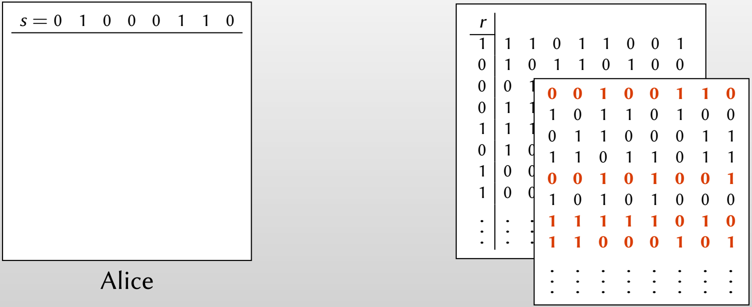
- Bob has input $r \Rightarrow$ extend to matrix

IKNP protocol [IshaiKilianNissimPetrank03]



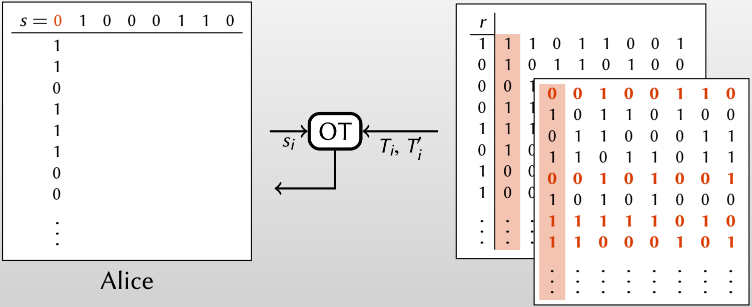
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')

IKNP protocol [IshaiKilianNissimPetrank03]



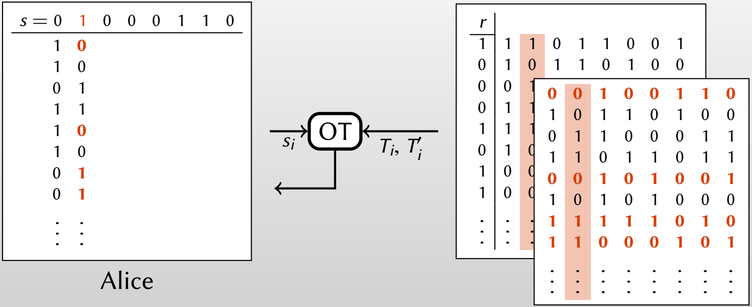
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s

IKNP protocol [IshaiKilianNissimPetrank03]



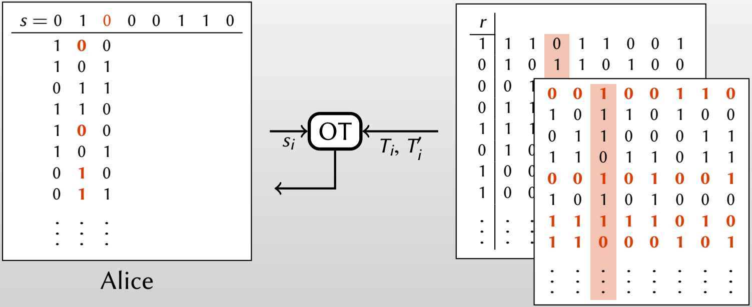
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

IKNP protocol [IshaiKilianNissimPetrank03]



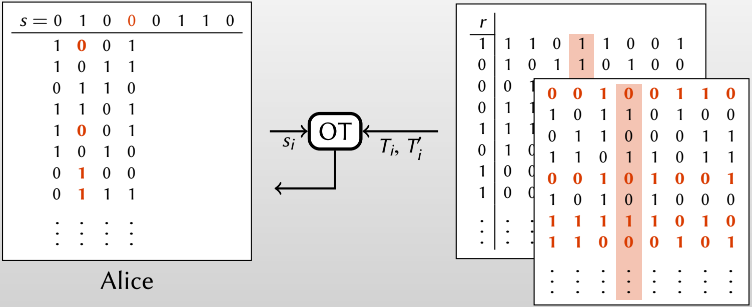
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

IKNP protocol [IshaiKilianNissimPetrank03]



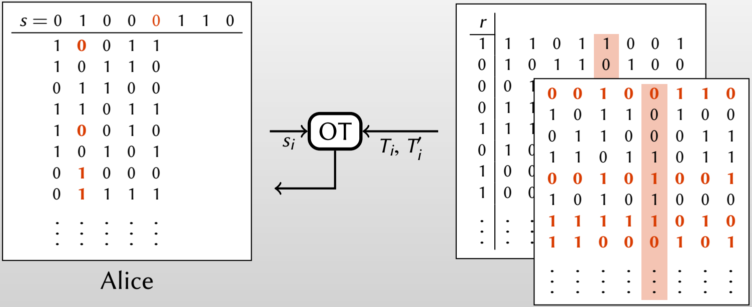
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

IKNP protocol [IshaiKilianNissimPetrank03]



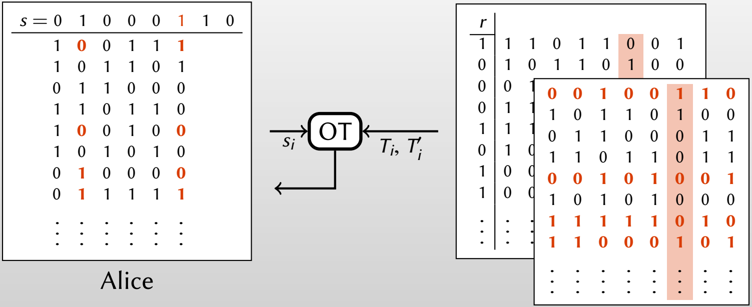
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

IKNP protocol [IshaiKilianNissimPetrank03]



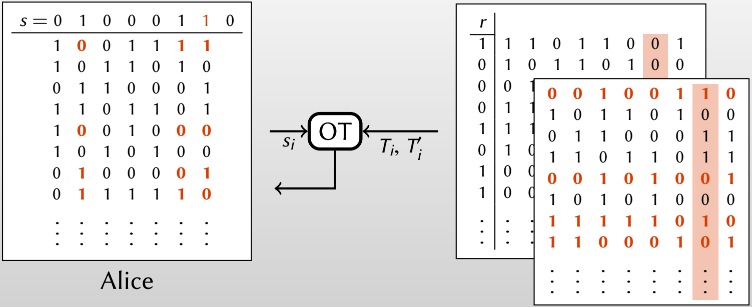
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

IKNP protocol [IshaiKilianNissimPetrank03]



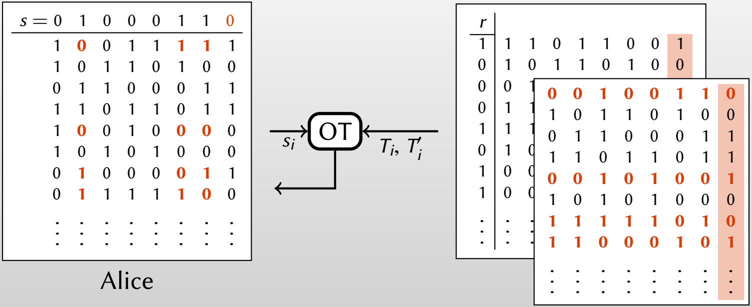
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

IKNP protocol [IshaiKilianNissimPetrank03]



- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

IKNP protocol [IshaiKilianNissimPetrank03]



- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

IKNP protocol [IshaiKilianNissimPetrank03]

$s =$ 0	1	0	0	0	1	1	0
1	**0**	0	1	1	**1**	**1**	1
1	0	1	1	0	1	0	0
0	1	1	0	0	0	1	1
1	1	0	1	1	0	1	1
1	**0**	0	1	0	**0**	**0**	0
1	0	1	0	1	0	0	0
0	**1**	0	0	0	**0**	**1**	1
0	**1**	1	1	1	**1**	**0**	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Alice

r								
1	1	1	0	1	1	0	0	1
0	1	0	1	1	0	1	0	0
0	0	1	1	0	0	0	1	1
0	1	1	0	1	1	0	1	1
1	1	1	0	1	0	1	1	0
0	1	0	1	0	1	0	0	0
1	0	0	0	0	0	1	0	1
1	0	0	1	1	1	0	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Bob

- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q
- Whenever $r_i = 0$, Alice row = Bob row

IKNP protocol [IshaiKilianNissimPetrank03]

$s =$ 0	1	0	0	0	1	1	0
1	0	0	1	1	1	1	1
1	0	1	1	0	1	0	0
0	1	1	0	0	0	1	1
1	1	0	1	1	0	1	1
1	0	0	1	0	0	0	0
1	0	1	0	1	0	0	0
0	1	0	0	0	0	1	1
0	1	1	1	1	1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

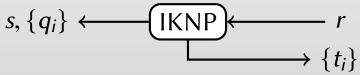
Alice

r								
1	1	1	0	1	1	0	0	1
0	1	0	1	1	0	1	0	0
0	0	1	1	0	0	0	1	1
0	1	1	0	1	1	0	1	1
1	1	1	0	1	0	1	1	0
0	1	0	1	0	1	0	0	0
1	0	0	0	0	0	1	0	1
1	0	0	1	1	1	0	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Bob

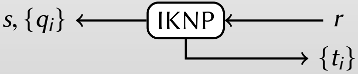
- Bob has input $r \Rightarrow$ extend to matrix and secret share as (T, T')
- Alice chooses random string s
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q
- Whenever $r_i = 0$, Alice row = Bob row
- Whenever $r_i = 1$, Alice row = Bob row $\oplus\, s$

IKNP protocol [IshaiKilianNissimPetrank03]



IKNP protocol [IshaiKilianNissimPetrank03]

q_1	$q_1 \oplus s$
q_2	$q_2 \oplus s$
q_3	$q_3 \oplus s$
$\vdots$	$\vdots$



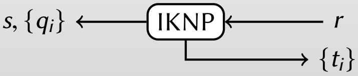
$r_1 = 0$	t_1
$r_2 = 1$	t_2
$r_3 = 1$	t_3
	$\vdots$

- For every i: Bob knows t_i; Alice knows q_i and $q_i \oplus s$

IKNP protocol [IshaiKilianNissimPetrank03]

$$q_i = \begin{cases} t_i & \text{if } r_i = 0 \\ t_i \oplus s & \text{if } r_i = 1 \end{cases}$$

t_1	$t_1 \oplus s$
$t_2 \oplus s$	t_2
$t_3 \oplus s$	t_3
⋮	⋮



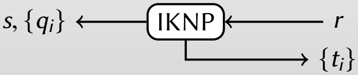
$r_1 = 0$	t_1
$r_2 = 1$	t_2
$r_3 = 1$	t_3
	⋮

- For every i: Bob knows t_i; Alice knows q_i and $q_i \oplus s$
- From Bob's perspective, he knows **exactly one** of Alice's two values: (Almost) an OT instance for each i!

IKNP protocol [IshaiKilianNissimPetrank03]

$$q_i = \begin{cases} t_i & \text{if } r_i = 0 \\ t_i \oplus s & \text{if } r_i = 1 \end{cases}$$

t_1	$t_1 \oplus s$
$t_2 \oplus s$	t_2
$t_3 \oplus s$	t_3
$\vdots$	$\vdots$



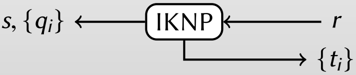
$r_1 = 0$	t_1
$r_2 = 1$	t_2
$r_3 = 1$	t_3
	$\vdots$

- For every i: Bob knows t_i; Alice knows q_i and $q_i \oplus s$
- From Bob's perspective, he knows **exactly one** of Alice's two values: (Almost) an OT instance for each i!
    - Reusing s leads to linear correlations in OT strings

IKNP protocol [IshaiKilianNissimPetrank03]

$$q_i = \begin{cases} t_i & \text{if } r_i = 0 \\ t_i \oplus s & \text{if } r_i = 1 \end{cases}$$

$H(t_1)$	$H(t_1 \oplus s)$
$H(t_2 \oplus s)$	$H(t_2)$
$H(t_3 \oplus s)$	$H(t_3)$
⋮	⋮

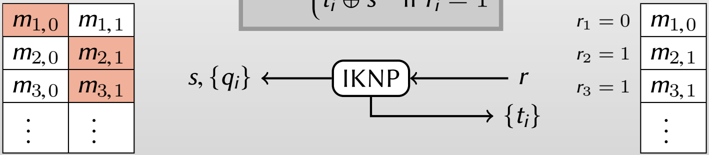


$r_1 = 0$	$H(t_1)$
$r_2 = 1$	$H(t_2)$
$r_3 = 1$	$H(t_3)$
	⋮

- For every i: Bob knows t_i; Alice knows q_i and $q_i \oplus s$
- From Bob's perspective, he knows **exactly one** of Alice's two values: (Almost) an OT instance for each i!
  - Reusing s leads to linear correlations in OT strings
- Break correlations by applying random oracle:
  - $H(t_1 \oplus s), \ldots H(t_n \oplus s)$ pseudorandom given $t_1, \ldots, t_n$ (secret s)

IKNP protocol [IshaiKilianNissimPetrank03]

$$q_i = \begin{cases} t_i & \text{if } r_i = 0 \\ t_i \oplus s & \text{if } r_i = 1 \end{cases}$$

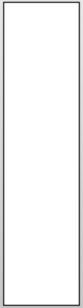


- For every i: Bob knows t_i; Alice knows q_i and $q_i \oplus s$
- From Bob's perspective, he knows **exactly one** of Alice's two values: (Almost) an OT instance for each i!
  - Reusing s leads to linear correlations in OT strings
- Break correlations by applying random oracle:
  - $H(t_1 \oplus s), \ldots H(t_n \oplus s)$ pseudorandom given $t_1, \ldots, t_n$ (secret s)

$\Rightarrow$ Random OT instance for each **row**, using base OT for each **column**

IKNP overview [IshaiKilianNissimPetrank03]

Tall matrices (λ columns, $n \gg \lambda$ rows)

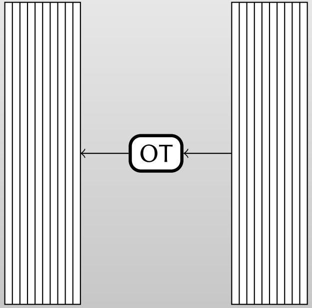


IKNP overview [IshaiKilianNissimPetrank03]

Tall matrices (λ columns, $n \gg \lambda$ rows)

Base OTs by column

- λ base OT instances
- transfer of n-bit strings



IKNP overview [IshaiKilianNissimPetrank03]

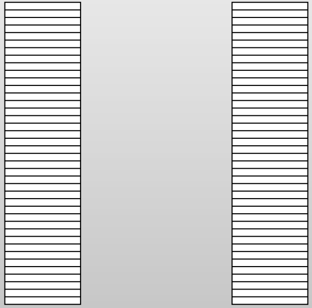
Tall matrices (λ columns, $n \gg \lambda$ rows)

Base OTs by column

- λ base OT instances
- transfer of n-bit strings

Obtain extended OT instance by row

- 1-2 evaluations of H per row



Generalizing IKNP [KolesnikovKumaresan13]

r
1
0
0
0
1
0
1
1
$\vdots$

- IKNP says: "Bob has r "

Generalizing IKNP [KolesnikovKumaresan13]

r								
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- IKNP says: “Bob has $r \Rightarrow$ extend to a matrix ”

Generalizing IKNP [KolesnikovKumaresan13]

$$
\begin{array}{c|cccccccc}
r & & & & & & & & \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
=
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
\oplus
\begin{array}{cccccccc}
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\
\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
$$

- IKNP says: “Bob has $r \Rightarrow$ extend to a matrix $\Rightarrow$ secret-share”

Generalizing IKNP [KolesnikovKumaresan13]

r								
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

=

1	1	0	1	1	0	0	1
1	0	1	1	0	1	0	0
0	1	1	0	0	0	1	1
1	1	0	1	1	0	1	1
1	1	0	1	0	1	1	0
1	0	1	0	1	0	0	0
0	0	0	0	0	1	0	1
0	0	1	1	1	0	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

⊕

0	**0**	**1**	**0**	**0**	**1**	**1**	**0**
1	0	1	1	0	1	0	0
0	1	1	0	0	0	1	1
1	1	0	1	1	0	1	1
0	**0**	**1**	**0**	**1**	**0**	**0**	**1**
1	0	1	0	1	0	0	0
1	**1**	**1**	**1**	**1**	**0**	**1**	**0**
1	**1**	**0**	**0**	**0**	**1**	**0**	**1**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- IKNP says: "Bob has $r \Rightarrow$ extend to a matrix $\Rightarrow$ secret-share"
- KK13 says: $0 \mapsto 000\cdots; 1 \mapsto 111\cdots$ is simple **repetition code**

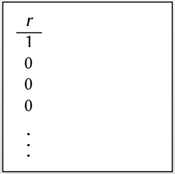
Generalizing IKNP [KolesnikovKumaresan13]

$$
\begin{array}{c|cccccccc}
r & & & & & & & & \\
\hline
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
=
\begin{array}{cccccccc}
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
\oplus
\begin{array}{cccccccc}
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\
\mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
$$

- IKNP says: "Bob has $r \Rightarrow$ extend to a matrix $\Rightarrow$ secret-share"
- KK13 says: $0 \mapsto 000\cdots; 1 \mapsto 111\cdots$ is simple **repetition code**

- **Generalize** by using a different error-correcting code.
  - Q: How do code properties (rate, distance) affect protocol?

Coding view of IKNP:



Bob

- Bob has input r

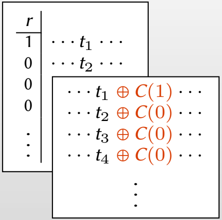
Coding view of IKNP:

r	
1	$\cdots C(1) \cdots$
0	$\cdots C(0) \cdots$
0	$\cdots C(0) \cdots$
0	$\cdots C(0) \cdots$
$\vdots$	$\vdots$

Bob

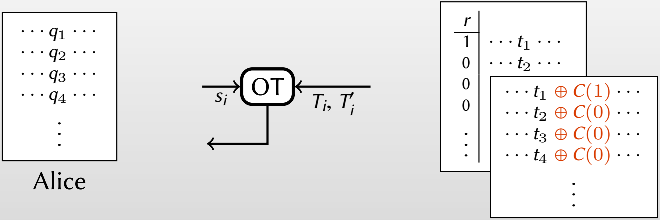
- Bob has input $r \Rightarrow$ **encode under C**

Coding view of IKNP:



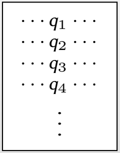
- Bob has input $r \Rightarrow$ **encode under C** and secret share as (T, T')

Coding view of IKNP:



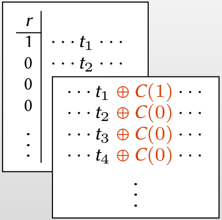
- Bob has input $r \Rightarrow$ **encode under C** and secret share as (T, T')
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

Coding view of IKNP:



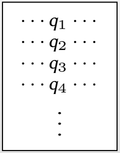
Alice

$$t_i = q_i \oplus C(r_i) \wedge s$$



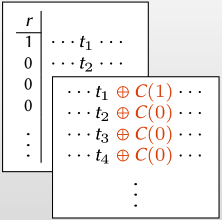
- Bob has input $r \Rightarrow$ **encode under C** and secret share as (T, T')
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

Coding view of IKNP:



Alice

$$t_i = q_i \oplus C(r_i) \wedge s$$

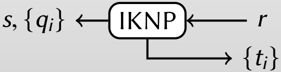


- Bob has input $r \Rightarrow$ **encode under C** and secret share as (T, T')
- OT for each **column** $\Rightarrow$ Alice obtains matrix Q

- Sanity check (using repetition code):

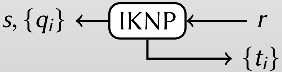
$$r_i = 0 \quad \Rightarrow \quad t_i = q_i \oplus (000\cdots) \wedge s = q_i$$
$$r_i = 1 \quad \Rightarrow \quad t_i = q_i \oplus (111\cdots) \wedge s = q_i \oplus s$$

Coding view of IKNP:



Coding view of IKNP:

$q_1 \oplus C(0) \wedge s$	$q_1 \oplus C(1) \wedge s$
$q_2 \oplus C(0) \wedge s$	$q_2 \oplus C(1) \wedge s$
$q_3 \oplus C(0) \wedge s$	$q_3 \oplus C(1) \wedge s$
$\vdots$	$\vdots$



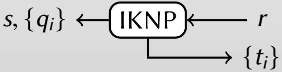
$r_1 = 0$	t_1
$r_2 = 1$	t_2
$r_3 = 1$	t_3
	$\vdots$

- For every i: Bob knows t_i; Alice knows $q_i \oplus C(0) \wedge s$ and $q_i \oplus C(1) \wedge s$

Coding view of IKNP:

$$t_i = q_i \oplus C(r_i) \wedge s$$

$t_1 \oplus C(0) \wedge s \oplus C(0) \wedge s$	$t_1 \oplus C(0) \wedge s \oplus C(1) \wedge s$
$t_2 \oplus C(1) \wedge s \oplus C(0) \wedge s$	$t_2 \oplus C(1) \wedge s \oplus C(1) \wedge s$
$t_3 \oplus C(1) \wedge s \oplus C(0) \wedge s$	$t_3 \oplus C(1) \wedge s \oplus C(1) \wedge s$
$\vdots$	$\vdots$



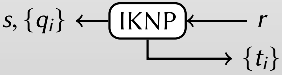
$r_1 = 0$	t_1
$r_2 = 1$	t_2
$r_3 = 1$	t_3
	$\vdots$

- For every i: Bob knows t_i; Alice knows $q_i \oplus C(0) \wedge s$ and $q_i \oplus C(1) \wedge s$
- Rewrite from Bob's point of view

Coding view of IKNP:

$$t_i = q_i \oplus C(r_i) \wedge s$$

$t_1 \oplus C(0 \oplus 0) \wedge s$	$t_1 \oplus C(0 \oplus 1) \wedge s$
$t_2 \oplus C(1 \oplus 0) \wedge s$	$t_2 \oplus C(1 \oplus 1) \wedge s$
$t_3 \oplus C(1 \oplus 0) \wedge s$	$t_3 \oplus C(1 \oplus 1) \wedge s$
$\vdots$	$\vdots$



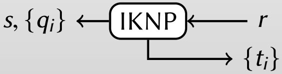
$r_1 = 0$	t_1
$r_2 = 1$	t_2
$r_3 = 1$	t_3
	$\vdots$

- For every i: Bob knows t_i; Alice knows $q_i \oplus C(0) \wedge s$ and $q_i \oplus C(1) \wedge s$
- Rewrite from Bob's point of view
- When C is a **linear code**: $[C(a) \wedge s] \oplus [C(b) \wedge s] = C(a \oplus b) \wedge s$

Coding view of IKNP:

$$t_i = q_i \oplus C(r_i) \wedge s$$

t_1	$t_1 \oplus C(1) \wedge s$
$t_2 \oplus C(1) \wedge s$	t_2
$t_3 \oplus C(1) \wedge s$	t_3
$\vdots$	$\vdots$



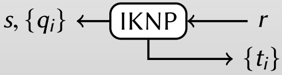
$r_1 = 0$	t_1
$r_2 = 1$	t_2
$r_3 = 1$	t_3
	$\vdots$

- For every i: Bob knows t_i; Alice knows $q_i \oplus C(0) \wedge s$ and $q_i \oplus C(1) \wedge s$
- Rewrite from Bob's point of view
- When C is a **linear code**: $[C(a) \wedge s] \oplus [C(b) \wedge s] = C(a \oplus b) \wedge s$ and $C(0) \wedge s = 00\cdots$

Coding view of IKNP:

$$t_i = q_i \oplus C(r_i) \wedge s$$

$H(t_1)$	$H(t_1 \oplus C(1) \wedge s)$
$H(t_2 \oplus C(1) \wedge s)$	$H(t_2)$
$H(t_3 \oplus C(1) \wedge s)$	$H(t_3)$
$\vdots$	$\vdots$

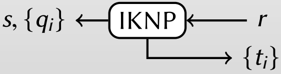


$r_1 = 0$	$H(t_1)$
$r_2 = 1$	$H(t_2)$
$r_3 = 1$	$H(t_3)$
	$\vdots$

- For every i: Bob knows t_i; Alice knows $q_i \oplus C(0) \wedge s$ and $q_i \oplus C(1) \wedge s$
- Rewrite from Bob's point of view
- When C is a **linear code**: $[C(a) \wedge s] \oplus [C(b) \wedge s] = C(a \oplus b) \wedge s$ and $C(0) \wedge s = 00\cdots$
- Use random oracle to destroy correlations

Generalizing IKNP:

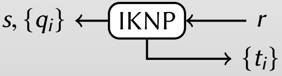
Consider a code that encodes more bits $C : \{0,1\}^3 \to \{0,1\}^k$



Generalizing IKNP:

Consider a code that encodes more bits $C : \{0,1\}^3 \to \{0,1\}^k$

$q_1 \oplus C(000) \wedge s$	$\cdots$	$q_1 \oplus C(111) \wedge s$
$q_2 \oplus C(000) \wedge s$	$\cdots$	$q_2 \oplus C(111) \wedge s$
$q_3 \oplus C(000) \wedge s$	$\cdots$	$q_3 \oplus C(111) \wedge s$
$\vdots$	$\vdots$	$\vdots$



t_1
t_2
t_3
$\vdots$

- For every *i*: Alice can compute (8 things)

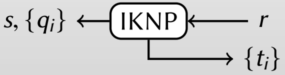
$$q_i \oplus C(000) \wedge s, \quad q_i \oplus C(001) \wedge s, \quad \ldots \quad q_i \oplus C(111) \wedge s$$

Generalizing IKNP:

Consider a code that encodes more bits $C : \{0,1\}^3 \rightarrow \{0,1\}^k$

$$t_i = q_i \oplus C(r_i) \wedge s$$

$t_1 \oplus C(r_1 \oplus 000) \wedge s$	$\cdots$	$t_1 \oplus C(r_1 \oplus 111) \wedge s$
$t_2 \oplus C(r_2 \oplus 000) \wedge s$	$\cdots$	$t_2 \oplus C(r_2 \oplus 111) \wedge s$
$t_3 \oplus C(r_3 \oplus 111) \wedge s$	$\cdots$	$t_3 \oplus C(r_3 \oplus 111) \wedge s$
$\vdots$	$\vdots$	$\vdots$



t_1
t_2
t_3
$\vdots$

- For every i: Alice can compute (8 things)

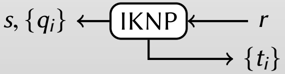
$$q_i \oplus C(000) \wedge s, \quad q_i \oplus C(001) \wedge s, \quad \ldots \quad q_i \oplus C(111) \wedge s$$

Generalizing IKNP:

Consider a code that encodes more bits $C : \{0,1\}^3 \rightarrow \{0,1\}^k$

$$t_i = q_i \oplus C(r_i) \wedge s$$

$t_1 \oplus C(r_1 \oplus 000) \wedge s$	$\cdots$	$t_1 \oplus C(r_1 \oplus 111) \wedge s$
$t_2 \oplus C(r_2 \oplus 000) \wedge s$	$\cdots$	$t_2 \oplus C(r_2 \oplus 111) \wedge s$
$t_3 \oplus C(r_3 \oplus 111) \wedge s$	$\cdots$	$t_3 \oplus C(r_3 \oplus 111) \wedge s$
$\vdots$	$\vdots$	$\vdots$



t_1
t_2
t_3
$\vdots$

- For every i: Alice can compute (8 things)

$$q_i \oplus C(000) \wedge s, \quad q_i \oplus C(001) \wedge s, \quad \ldots \quad q_i \oplus C(111) \wedge s$$

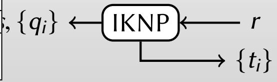
- Bob knows exactly 1 of the 8 values (corresponding to r_i)
  - Others are of the form $t \oplus c \wedge s$ for known t and **codeword** c

Generalizing IKNP:

Consider a code that encodes more bits $C : \{0,1\}^3 \to \{0,1\}^k$

$$t_i = q_i \oplus C(r_i) \wedge s$$

$H(t_1 \oplus C(r_1 \oplus 000) \wedge s)$	$\cdots$	$H(t_1 \oplus C(r_1 \oplus 111) \wedge s)$
$H(t_2 \oplus C(r_2 \oplus 000) \wedge s)$	$\cdots$	$H(t_2 \oplus C(r_2 \oplus 111) \wedge s)$
$H(t_3 \oplus C(r_3 \oplus 000) \wedge s)$	$\cdots$	$H(t_3 \oplus C(r_3 \oplus 111) \wedge s)$
$\vdots$	$\vdots$	$\vdots$



$H(t_1)$
$H(t_2)$
$H(t_3)$
$\vdots$

- For every i: Alice can compute (8 things)

$$q_i \oplus C(000) \wedge s, \quad q_i \oplus C(001) \wedge s, \quad \ldots \quad q_i \oplus C(111) \wedge s$$

- Bob knows exactly 1 of the 8 values (corresponding to r_i)
  - Others are of the form $t \oplus c \wedge s$ for known t and **codeword** c
- In the random oracle model:
  - $H(t_1 \oplus c_1 \wedge s), \ldots H(t_n \oplus c_n \wedge s)$ pseudorandom **if all c_i have Hamming weight** $\geq \lambda$

Generalizing IKNP:

[KolesnikovKumaresan13]

Using a code $C : \{0,1\}^\ell \to \{0,1\}^k$ with minimum distance λ gives you 1-out-of-2^ℓ OT extension (from k base OTs)

[KolesnikovKumaresan13]:

- Walsh-Hadamard code $C : \{0,1\}^8 \to \{0,1\}^{256}$ (min. dist. 128)
- 1-out-of-256 OT

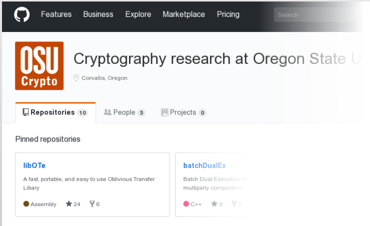
[OrruOrsiniScholl16]:

- BCH code $C : \{0,1\}^{76} \to \{0,1\}^{512}$ (min. dist. 171)
- 1-out-of-2^{76} OT

[KolesnikovKumaresanRosulekTrieu16]:

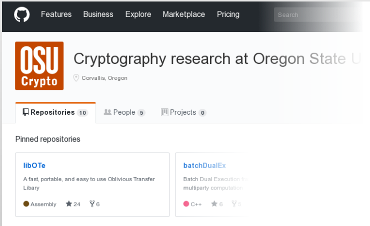
- Any pseudorandom function $C : \{0,1\}^* \to \{0,1\}^{\sim 480}$
- Linearity and decoding properties not needed (only min. dist. whp)!
- 1-out-of-∞ OT

Perspective



semi-honest	1-out-of-2	28 million / sec
malicious	1-out-of-2	24 million / sec
semi-honest	1-out-of-N	2.5 million / sec
malicious	1-out-of-N	1.8 million / sec

Perspective



semi-honest	1-out-of-2	28 million / sec
malicious	1-out-of-2	24 million / sec
semi-honest	1-out-of-N	2.5 million / sec
malicious	1-out-of-N	1.8 million / sec

OTs are cheap!