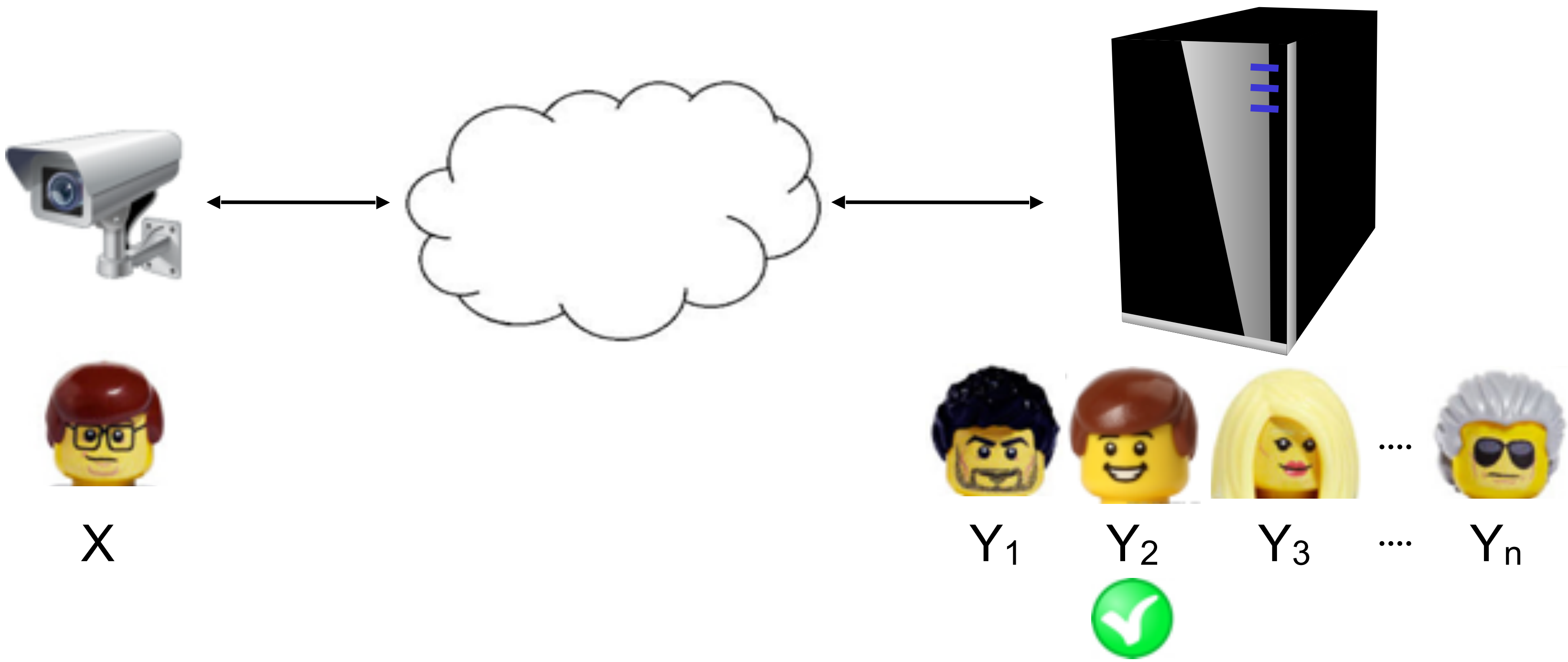
GSHADE: Faster Privacy-Preserving Distance Computation and Biometric Identification

Thomas Schneider (TU Darmstadt)

based on joint works with
Michael Zohner (TU Darmstadt)
Julien Bringer, Hervé Chabanne, Mélanie Favre, Alain Patey (Morpho)
Gilad Asharov, Yehuda Lindell (Bar-Ilan University)

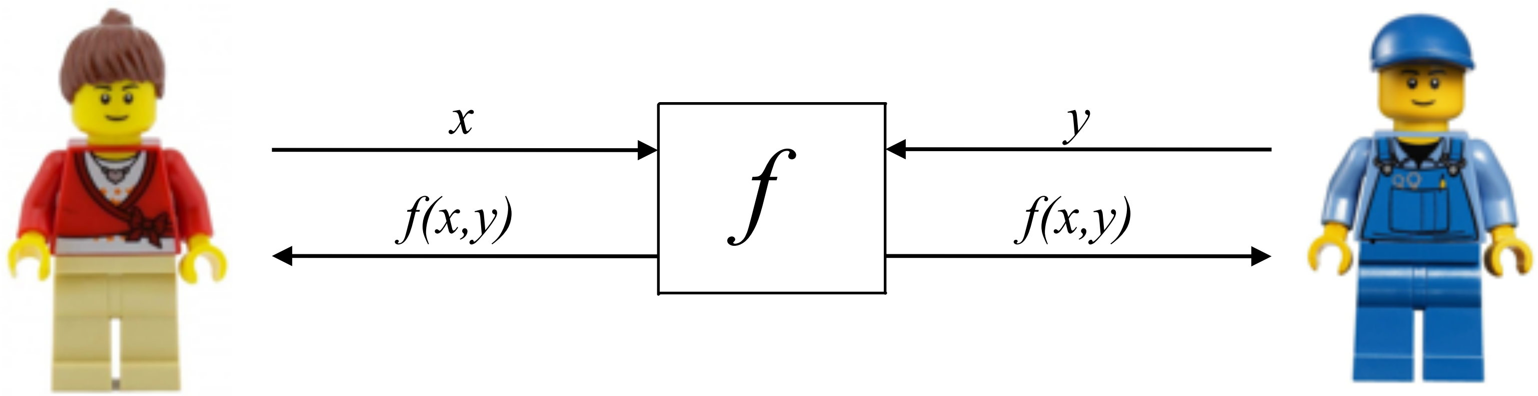
Workshop on PETs for Biometric Data, Haifa, Jan 15, 2015

Privacy-Preserving Biometric Identification



Task: Check if query is ***similar*** to an entry in the DB.
- without revealing the query to the server
- without revealing the DB to the client

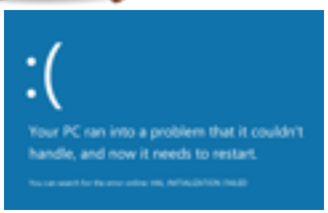
Secure Two-Party Computation



This Talk: **Passive** Adversaries

Example Privacy-Preserving Applications

Auctions [NaorPS99], ...



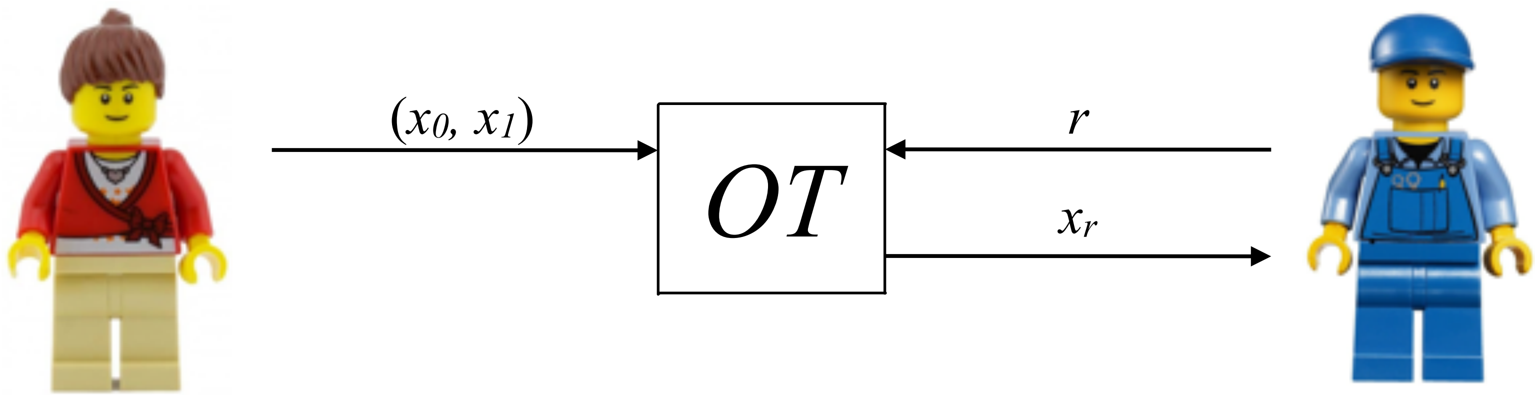
Remote Diagnostics [BrickellPSW07], ...

DNA Searching [Troncoso-PastorizaKC07], ...

Biometric Identification [ErkinFGKLT09], ...

Medical Diagnostics [BarniFKLSS09], ...

Oblivious Transfer (OT)



OT is fundament of many secure computation protocols.

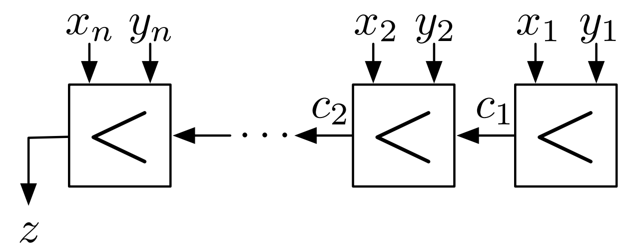
Yao's Garbled Circuits Protocol [Yao'86]

$f(\cdot,\cdot)$ e.g., $\mathbf{x} < \mathbf{y}$

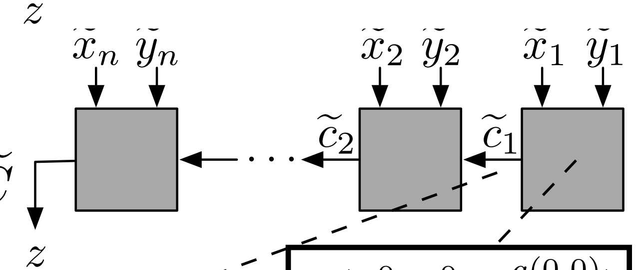
private data $\mathbf{x} = x_1, .., x_n$

private data $\mathbf{y} = y_1, .., y_n$

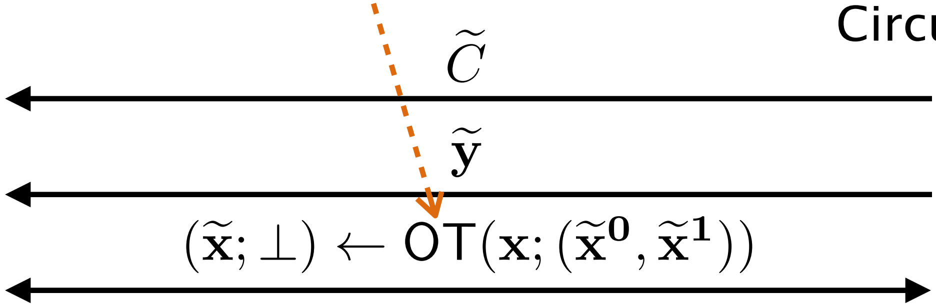
- Circuit



- Garbled Circuit $\widetilde{C}$



OT on keys per Alice's input bit



$f(\mathbf{x},\mathbf{y}) = \widetilde{C}(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}})$

$\widetilde{c}_1^0, \widetilde{c}_1^1$

Garbled Values

$E(\widetilde{x}_1^0, \widetilde{y}_1^0;\ \widetilde{c}_1^{g(0,0)})$

$E(\widetilde{x}_1^0, \widetilde{y}_1^1;\ \widetilde{c}_1^{g(0,1)})$

$E(\widetilde{x}_1^1, \widetilde{y}_1^0;\ \widetilde{c}_1^{g(1,0)})$

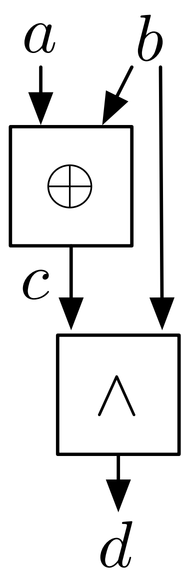
$E(\widetilde{x}_1^1, \widetilde{y}_1^1;\ \widetilde{c}_1^{g(1,1)})$

Garbled Table

The GMW Protocol
[Goldreich/Micali/Wigderson'87]

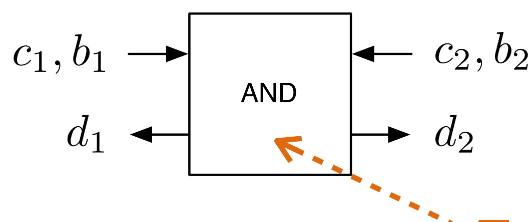
Secret share inputs: $a = a_1 \oplus a_2$

$b = b_1 \oplus b_2$



Non-Interactive XOR gates: $c_1 = a_1 \oplus b_1$; $c_2 = a_2 \oplus b_2$

Interactive AND gates:

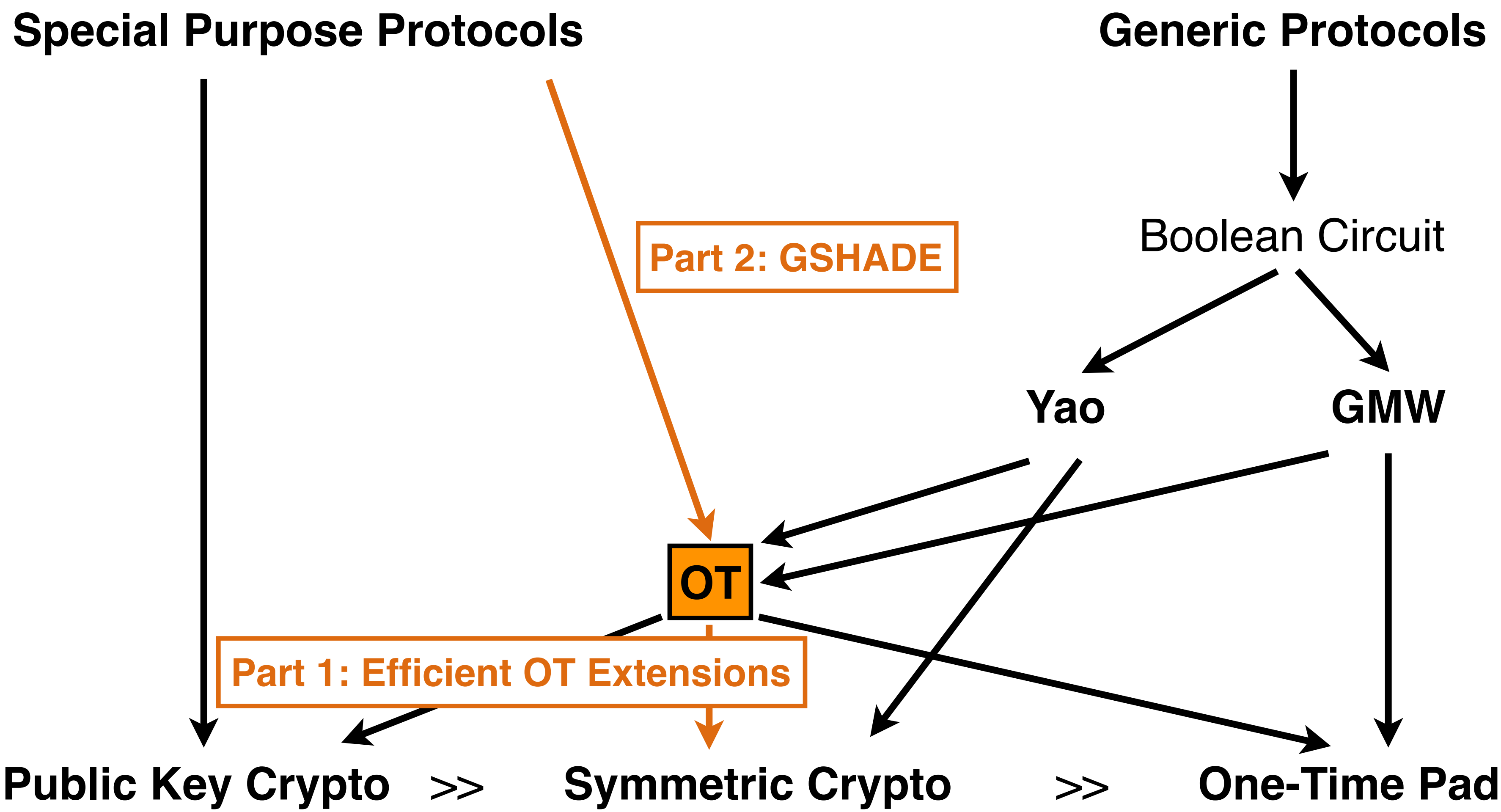


Two OTs on bits per AND gate

Recombine outputs: $d = d_1 \oplus d_2$

Overview of this talk: Secure Computation

TECHNISCHE UNIVERSITÄT DARMSTADT



EC SPRIDE

Part 1: Efficient OT Extensions

http://encrypto.de/code/OTExtension

G. Asharov, Y. Lindell, T. Schneider, M. Zohner:
More efficient oblivious transfer and extensions for faster secure computation.
In ACM CCS'13.

OT - Bad News

TECHNISCHE UNIVERSITÄT DARMSTADT

- [ImpagliazzoRudich’89]: there’s no black-box reduction from OT to OWFs

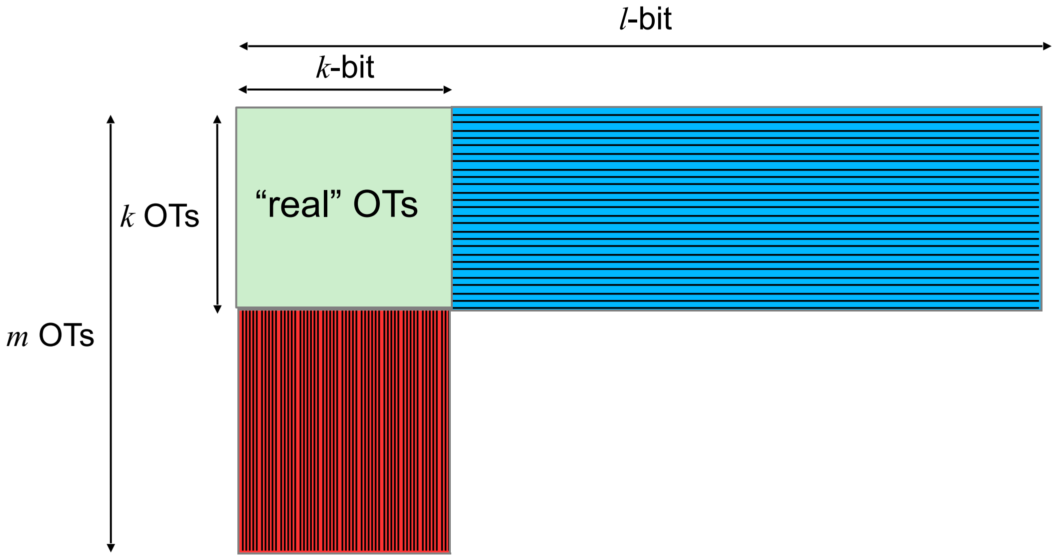
- Several OT protocols based on public-key cryptography
  - e.g., [NaorPinkas’01] yields ~1,000 OTs per second

- Since public-key crypto is expensive, OT was believed to be inefficient

OT - Good News

- [Beaver'95]: OTs can be pre-computed (only OTP in online phase)

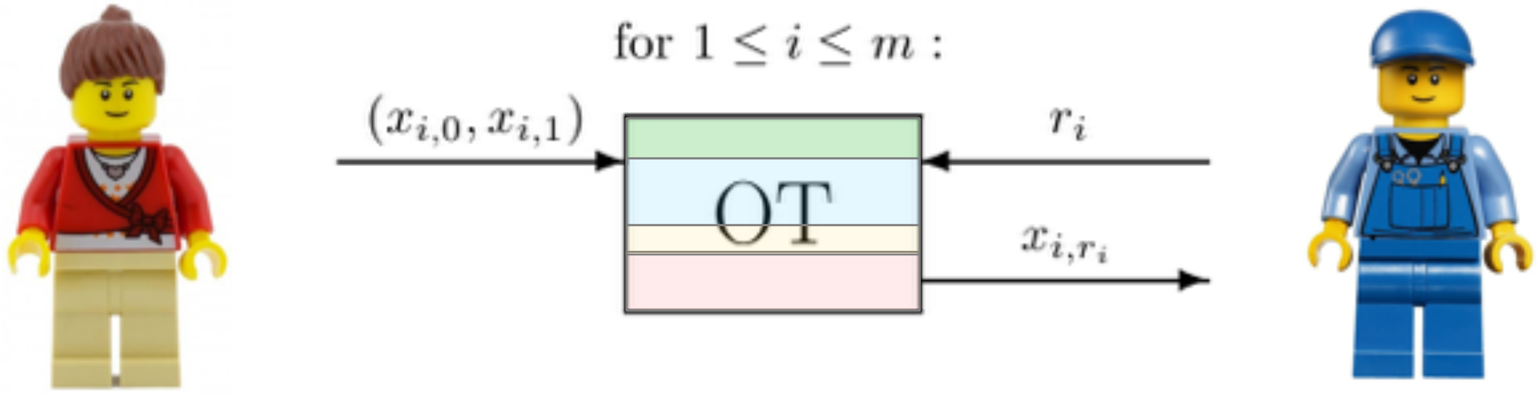
- OT Extensions (similar to hybrid encryption):
  use symmetric crypto to stretch few "real" OTs into longer/many OTs
    - [Beaver'96]: OT on long strings from short seeds
    - [IshaiKilianNissimPetrank'03]: many OTs from few OTs



OT Extension of [IKNP'03] (1)

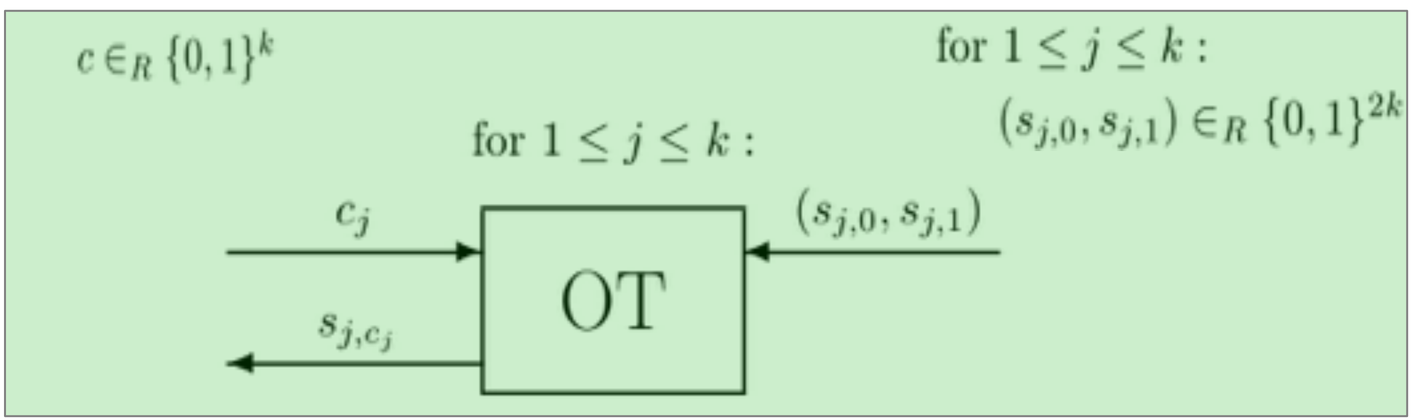
- Alice inputs m pairs of ℓ-bit pairs $(x_{i,0}, x_{i,1})$

- Bob inputs m-bit string r and obtains $x_{i,ri}$ in i-th OT



OT Extension of [IKNP'03] (2)

- Alice and Bob perform k "real" OTs on random seeds with reverse roles (k: security parameter)



OT Extension of [IKNP'03] (3)

- Bob generates a random m x k bit matrix $\mathbf{T}$ and masks his choices r

- The matrix is masked with the stretched seeds of the "real" OTs

$$\mathbf{T} \in_R \{0,1\}^{m \times k}$$

$$\text{for } 1 \le j \le k:$$

$$u_{j,0} = PRG(s_{j,0}) \oplus \mathbf{T}[\text{j}]$$

$$u_{j,1} = PRG(s_{j,1}) \oplus \mathbf{T}[\text{j}] \oplus \mathbf{r}$$

$$\xleftarrow{(u_{j,0}, u_{j,1}),\ 1 \le i \le k}$$

$$\text{for } 1 \le j \le k:$$

$$\mathbf{V}[\text{j}] = u_{j,c_j} \oplus PRG(s_{j,c_j})$$

PRG: pseudo-random generator (instantiated with AES)

OT Extension of [IKNP'03] (4)

- Transpose matrices **V** and **T**

- Alice masks her inputs and obliviously sends them to Bob

$\mathbf{V'} = \mathbf{V}^T$ $\qquad\qquad$ $\mathbf{T'} = \mathbf{T}^T$

for $1 \leq i \leq m$:

$$y_{i,0} = x_{i,0} \oplus H(i, \mathbf{V'}[\mathrm{i}])$$

$$y_{i,1} = x_{i,1} \oplus H(i, \mathbf{V'}[\mathrm{i}] \oplus c)$$

$$\xrightarrow{(y_{i,0}, y_{i,1}), 1 \leq i \leq m}$$

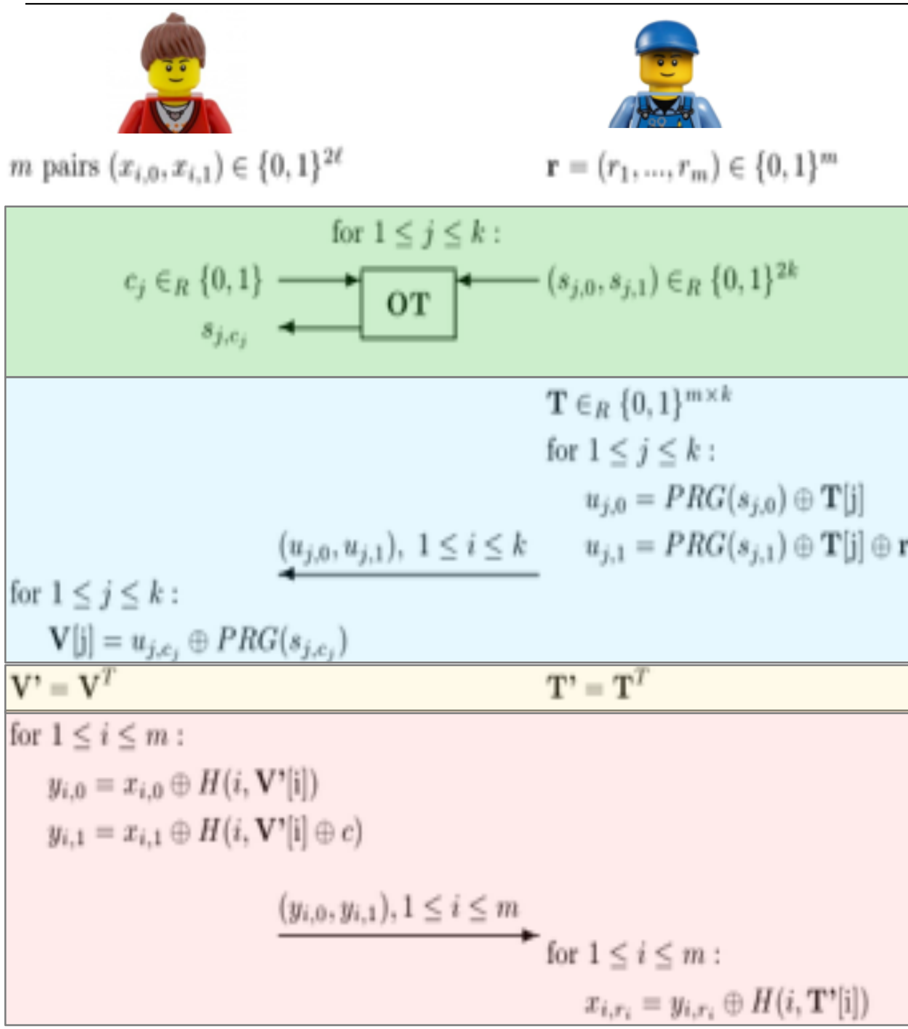
for $1 \leq i \leq m$:

$$x_{i,r_i} = y_{i,r_i} \oplus H(i, \mathbf{T'}[\mathrm{i}])$$

H: correlation robust function (instantiated with hash function)

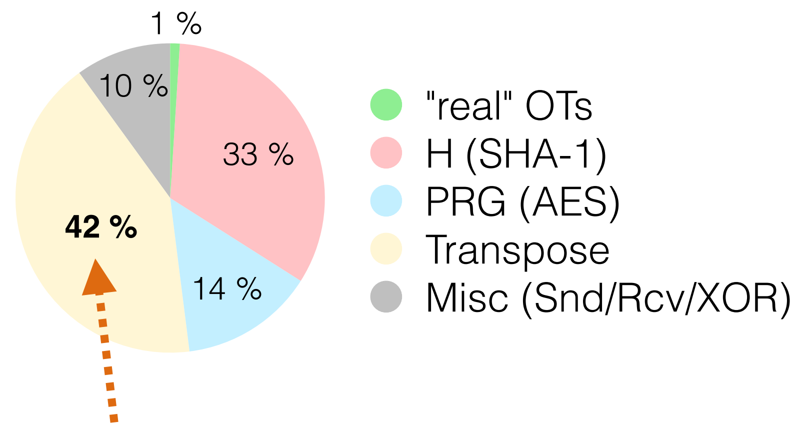
Computation Complexity of OT Extension

TECHNISCHE UNIVERSITÄT DARMSTADT



	Per OT:	
1	# PRG evaluations	2
2	# H evaluations	1

Time distribution for 10 Mio. OTs (in 21s):



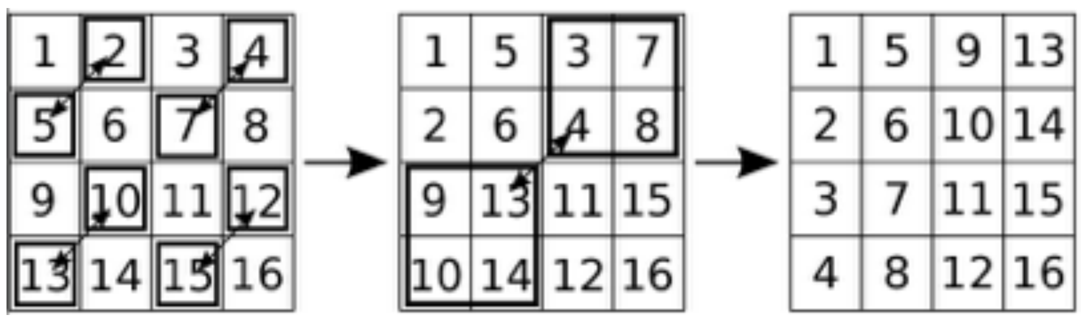
Non-crypto part is bottleneck!!!

Algorithmic Optimization
Efficient Bit-Matrix Transposition

TECHNISCHE UNIVERSITÄT DARMSTADT

- Naive matrix transposition performs mk load/process/store operations

- Eklundh's algorithm reduces number of operations to O($m \log_2 k$) swaps
  - Swap whole registers instead of bits
  - Transposing 10 times faster

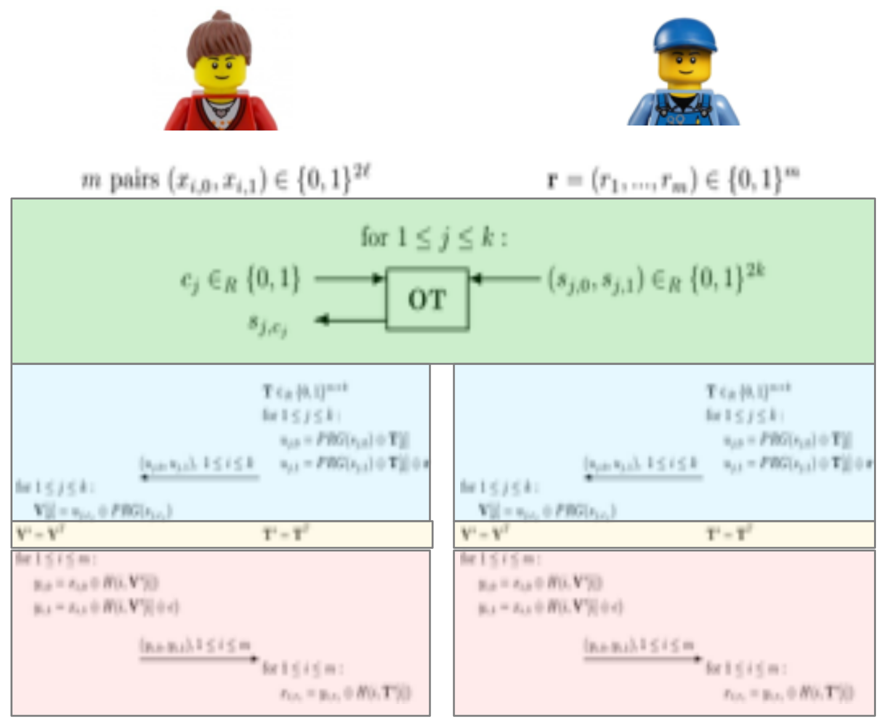


EC SPRIDE

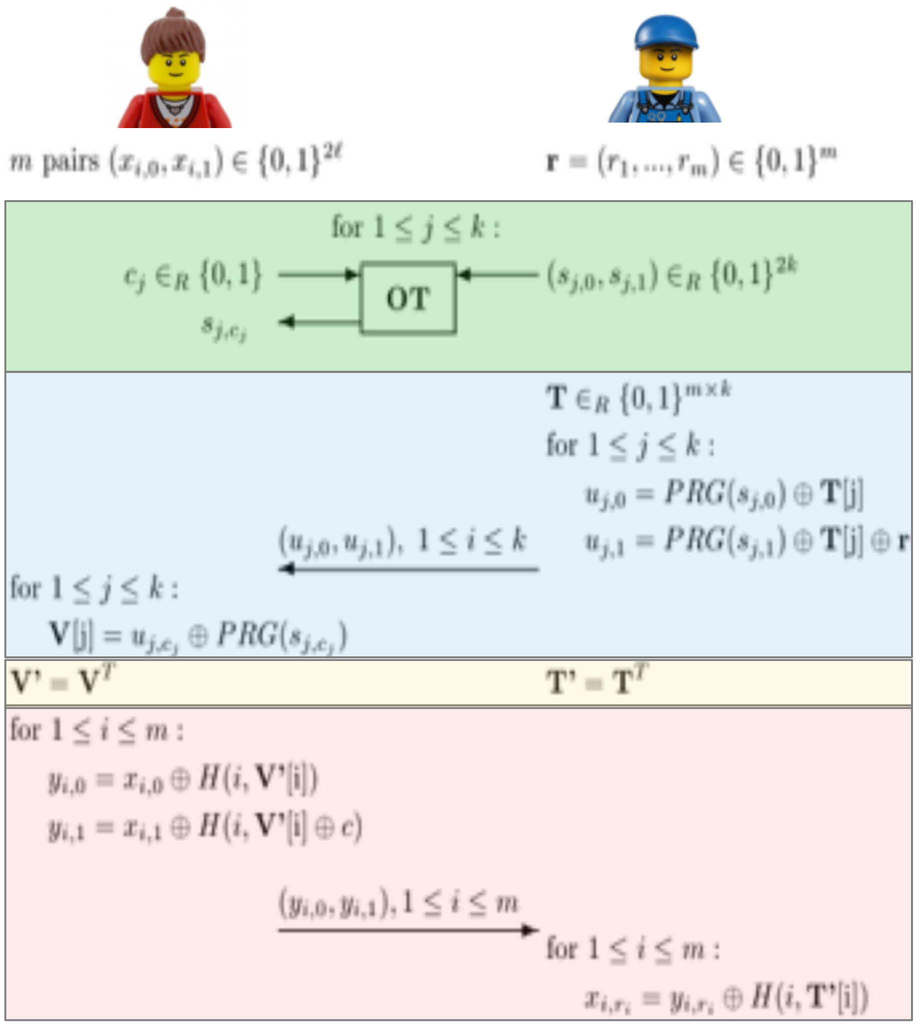
Algorithmic Optimization
Parallelized OT Extension

- OT extension can easily be parallelized by splitting the $\mathbf{T}$ matrix into sub-matrices

- Since columns are independent, OT is highly parallelizable



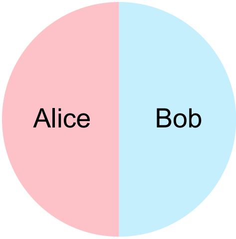
Communication Complexity of OT Extension



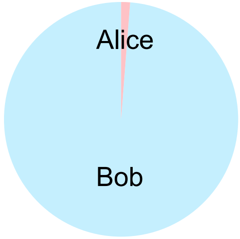
Per OT:

2ℓ	Bits sent	$2k$

Yao: $\ell = k = 80$



GMW: $\ell = 1,\ k = 80$



Protocol Optimization
General OT Extension

- Instead of generating a random T matrix, we derive it from $s_{j,0}$

- Reduces data sent by Bob by factor 2

$\mathbf{T} \in_R \{0,1\}^{m \times k}$

for $1 \leq j \leq k$:

$u_{j,0} = PRG(s_{j,0}) \oplus \mathbf{T}[j]$

$u_{j,1} = PRG(s_{j,1}) \oplus \mathbf{T}[j] \oplus \mathbf{r}$

$\xleftarrow{(u_{j,0}, u_{j,1}),\ 1 \leq i \leq k}$

for $1 \leq j \leq k$:

$\mathbf{V}[j] = u_{j,c_j} \oplus PRG(s_{j,c_j})$

for $1 \leq j \leq k$:

$\mathbf{T}[j] = PRG(s_{j,0})$

$u_j = PRG(s_{j,1}) \oplus \mathbf{T}[j] \oplus \mathbf{r}$

$\xleftarrow{u_j,\ 1 \leq i \leq k}$

for $1 \leq j \leq k$:

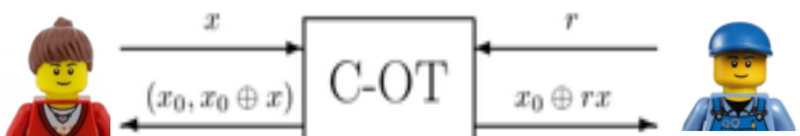
$\mathbf{V}[j] = c_j u_j \oplus PRG(s_{j,c_j})$

Specific OT Functionalities

TECHNISCHE UNIVERSITÄT DARMSTADT

- Secure computation protocols often require a specific OT functionality
  - Yao with free XORs requires strings x_0, x_1 to be XOR-correlated
  - GMW with multiplication triples can use random strings

- Correlated OT: random x_0 and $x_1 = x_0 \oplus x$



Correlated OT

e.g., for Yao

- Random OT: random x_0 and x_1

Random OT

e.g., for GMW

Specific OT Functionalities
Correlated OT Extension (C-OT)

- Choose $x_{i,0}$ as random output of H (modeled as RO here)

- Compute $x_{i,1}$ as $x_{i,0} \oplus x_i$ to obliviously transfer XOR-correlated values

- Reduces data sent by Alice by factor 2

$\mathbf{V'} = \mathbf{V}^T$ $\qquad$ $\mathbf{T'} = \mathbf{T}^T$

for $1 \leq i \leq m$:

$y_{i,0} = x_{i,0} \oplus H(i, \mathbf{V'}[i])$

$y_{i,1} = x_{i,1} \oplus H(i, \mathbf{V'}[i] \oplus c)$

$(y_{i,0}, y_{i,1}), 1 \leq i \leq m \longrightarrow$

for $1 \leq i \leq m$:

$x_{i,r_i} = y_{i,r_i} \oplus H(i, \mathbf{T'}[i])$

$\mathbf{V'} = \mathbf{V}^T$ $\qquad$ $\mathbf{T'} = \mathbf{T}^T$

for $1 \leq i \leq m$:

$x_{i,0} = H(i, \mathbf{V'}[i])$

$x_{i,1} = x_{i,0} \oplus x_i$

$y_i = x_{i,1} \oplus H(i, \mathbf{V'}[i] \oplus c)$

$y_i, 1 \leq i \leq m \longrightarrow$

for $1 \leq i \leq m$:

$x_{i,r_i} = r_i y_i \oplus H(i, \mathbf{T'}[i])$

Specific OT Functionalities
Random OT Extension (R-OT)

- Choose $x_{i,0}$ and $x_{i,1}$ as random outputs of H (modeled as RO here)

- No data sent by Alice


$\mathbf{V'} = \mathbf{V}^T$ $\quad\quad$ $\mathbf{T'} = \mathbf{T}^T$

for $1 \le i \le m$:

$y_{i,0} = x_{i,0} \oplus H(i, \mathbf{V'}[i])$

$y_{i,1} = x_{i,1} \oplus H(i, \mathbf{V'}[i] \oplus c)$

$\xrightarrow{(y_{i,0}, y_{i,1}), 1 \le i \le m}$ for $1 \le i \le m$:

$x_{i,r_i} = y_{i,r_i} \oplus H(i, \mathbf{T'}[i])$

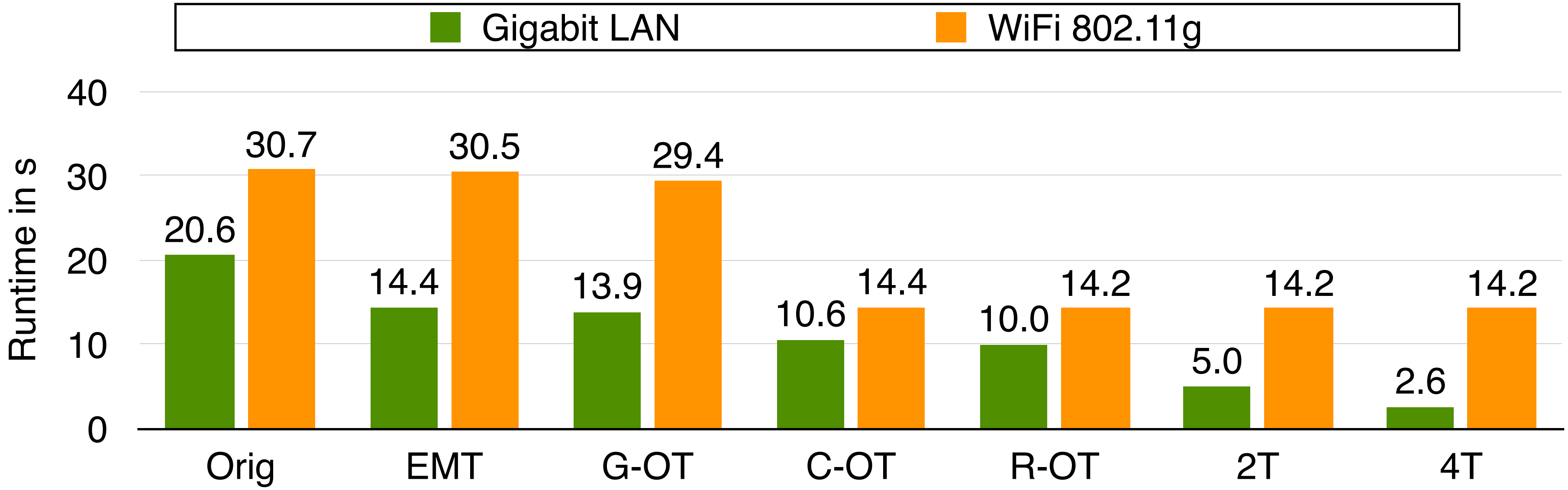

$\mathbf{V'} = \mathbf{V}^T$ $\quad\quad$ $\mathbf{T'} = \mathbf{T}^T$

for $1 \le i \le m$: $\quad\quad$ for $1 \le i \le m$:

$x_{i,0} = H(i, \mathbf{V'}[i])$ $\quad\quad$ $x_{i,r_i} = H(i, \mathbf{T'}[i])$

$x_{i,1} = H(i, \mathbf{V'}[i] \oplus c)$

Performance Evaluation
Conclusion



Performance for 10 Mio. OTs on 80-bit strings

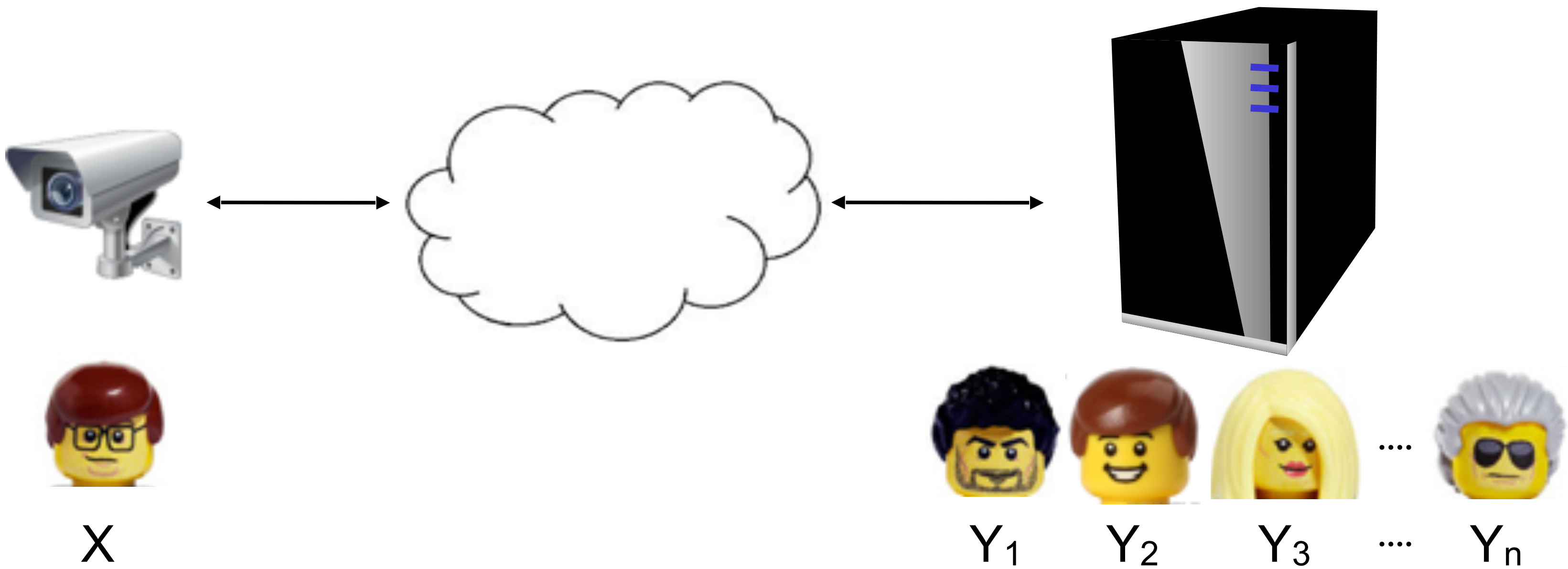
- OT is very efficient

- **Communication** is the **bottleneck** for OT (even without using AES-NI)

Part 2: GSHADE

J. Bringer, H. Chabanne, M. Favre, A. Patey, T. Schneider, M. Zohner*:*
GSHADE: Faster privacy-preserving distance computation and biometric identification.
In ACM IH&MMSEC'14.

Privacy-Preserving Biometric Identification



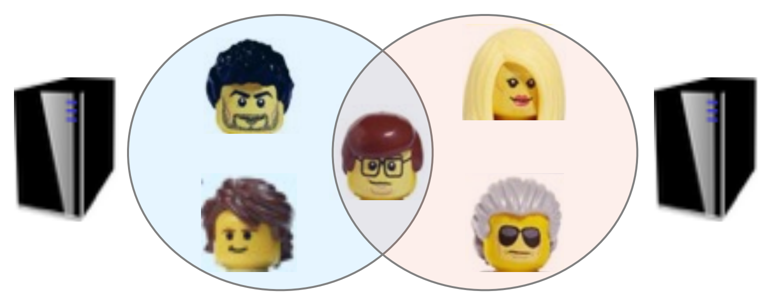
Task: Check if query is ***similar*** to an entry in the DB.
- without revealing the query to the server
- without revealing the DB to the client

Use-Cases

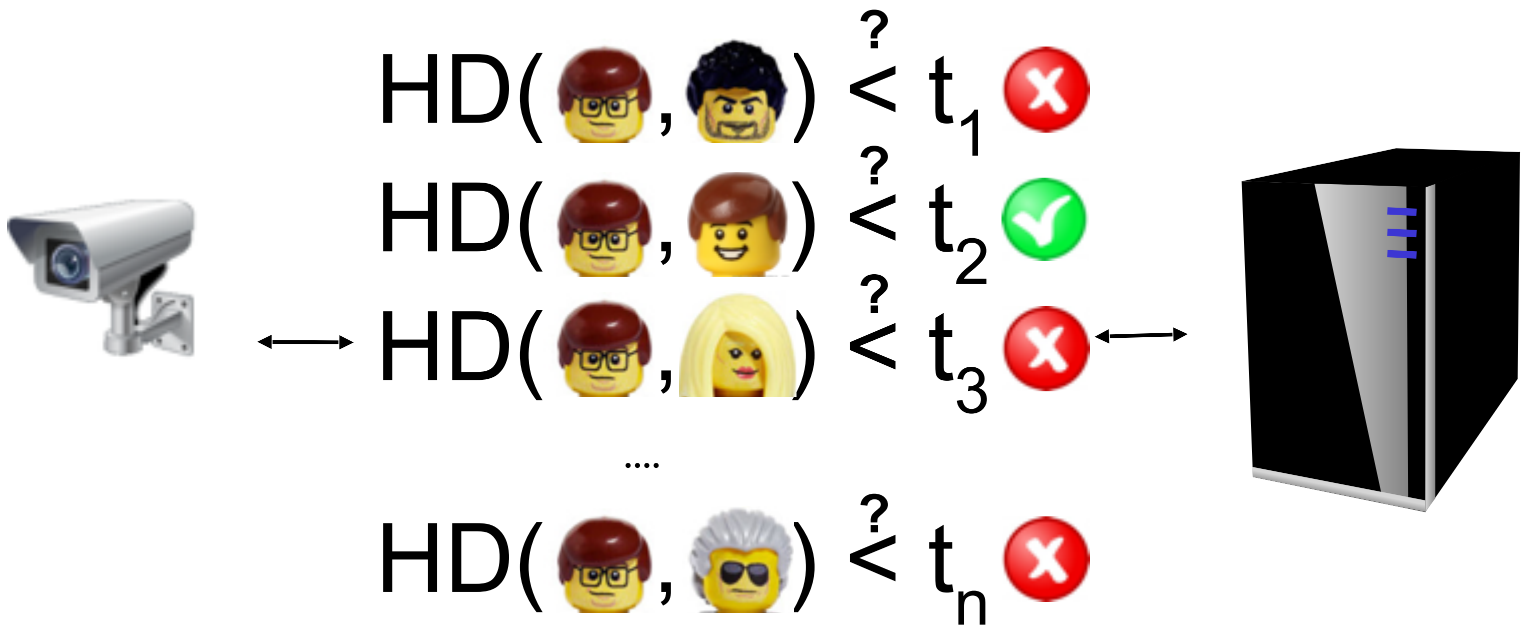
Biometric Access Control / Border Control

Anonymous Biometric Credentials

Secure Biometric Database Intersection



The SCiFI Algorithm
[Osadchy/Pinkas/Jarrous/Moskovich S&P'10]



Compute Hamming distance of ℓ=900 bit strings and compare with threshold.

Privacy-Preserving Biometric Identification: Classification

Technique / Distance	Public-Key Crypto	Boolean / Hybrid	OT-based
Hamming (HD) 	[OPJM10]	[HEKM11] [**S**Z13]	[BCP13] SHADE **GSHADE**
Euclidean  	[EFG+09]	[S**S**W09] [HKS+10] [BG11] [HMEK11] [**S**Z13]	**GSHADE**
Normalized HD 	-	[BG11]	**GSHADE**

SHADE

Secure Hamming Dist. computation from OT [BringerChabannePatey'13]

Goal: compute $HD(X,Y) = \Sigma(x_i \oplus y_i)$, $i=1..\ell$

for $i=1..\ell$:

choose $r_i \in_R \mathbb{Z}_{\ell+1}$

r_i+y_i ; $r_i+(1-y_i)$ → OT ← x_i

OT → $t_i=r_i+(x_i \oplus y_i)$

$R = \Sigma r_i$

$T = \Sigma t_i = R + HD(X,Y)$

Continue with generic MPC protocol (e.g., Yao or GMW) from $T - R = HD(X,Y)$ …

GSHADE: Optimizations and Generalization of SHADE

- For multiple HD computations: $HD(X,Y_1)$, $HD(X,Y_2)$, …:
  Same number of OTs, but on longer strings
- Can use correlated OT (C-OT) to improve communication
- Generalize to larger class of functions $f(X,Y) = f_X(X) + f_Y(Y) + \Sigma f_i(x_i,Y)$
  - Hamming Distance: $f_X=f_Y=0$, $f_i(x_i,Y)=x_i \oplus y_i$
  - Squared Euclidean Distance (for faces & fingerprints):
    $f_X(X)=\Sigma x_i^2$, $f_Y(Y)=\Sigma y_i^2$, $f_i(x_i,Y)=-2x_i y_i$
  - Normalized Hamming Distance (for irises) $\frac{\sum_{i=1}^{\ell}(m_i m'_i(x_i \oplus y_i))}{\sum_{i=1}^{\ell}(m_i m'_i)}$
  - Squared Mahalanobis Distance
    (for hand shapes, keystrokes, signatures) $(X-Y)^T M(X-Y)$

GSHADE Protocol

Goal: compute $f(X,Y) = f_X(X) + f_Y(Y) + \Sigma f_i(x_i,Y)$

for $i=1..\ell$:

choose $r_i \in_R \mathbb{Z}_m$

$r_i+f_i(0,Y);\ r_i+f_i(1,Y)$ → *OT* ← x_i

OT → $t_i=r_i+f_i(x_i,Y)$

$R = - f_Y(Y) + \Sigma r_i$

$T = f_X(X) + \Sigma t_i$

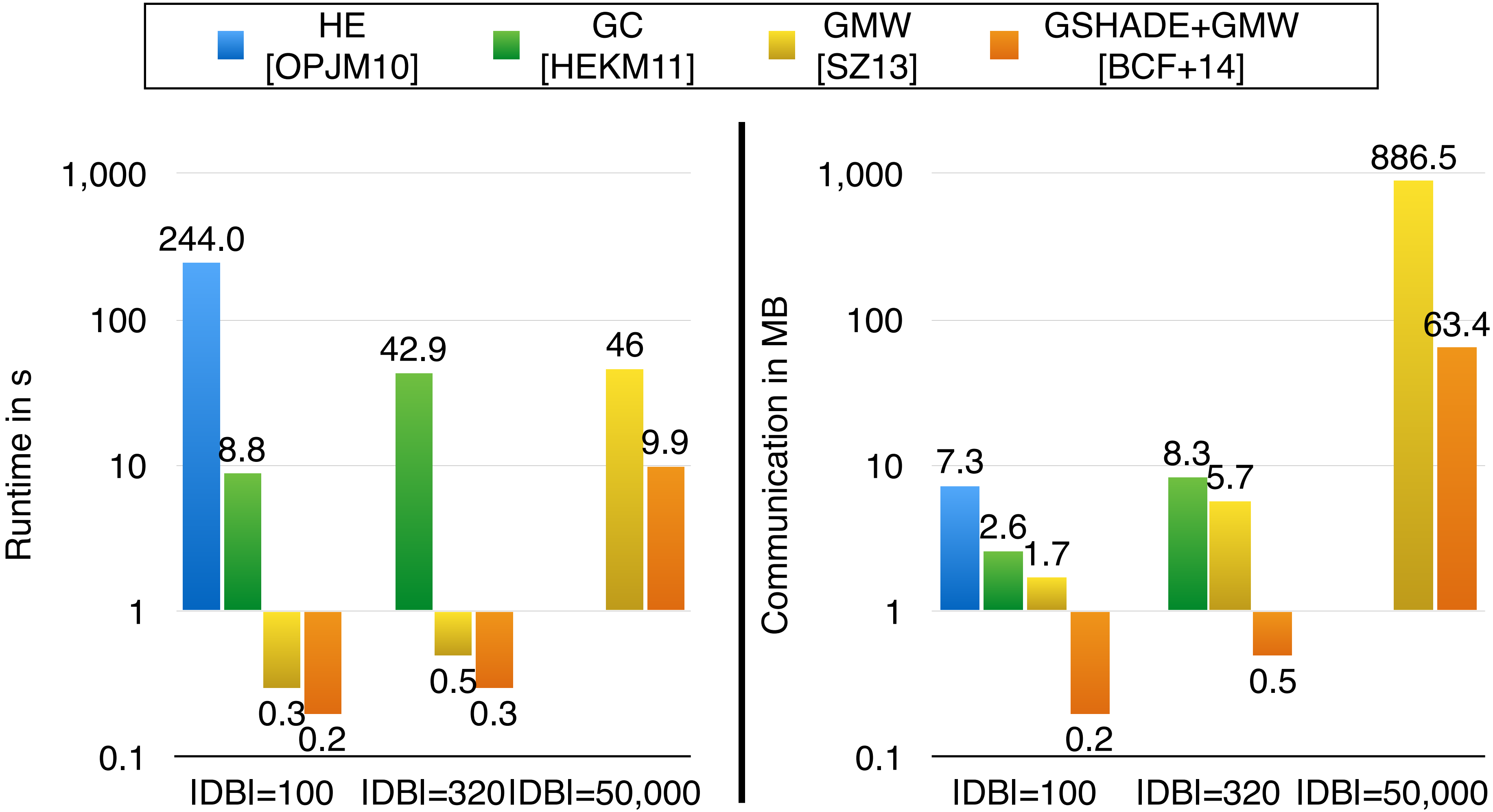
Continue with generic MPC from T - R = f(X,Y) = …

Performance of GSHADE

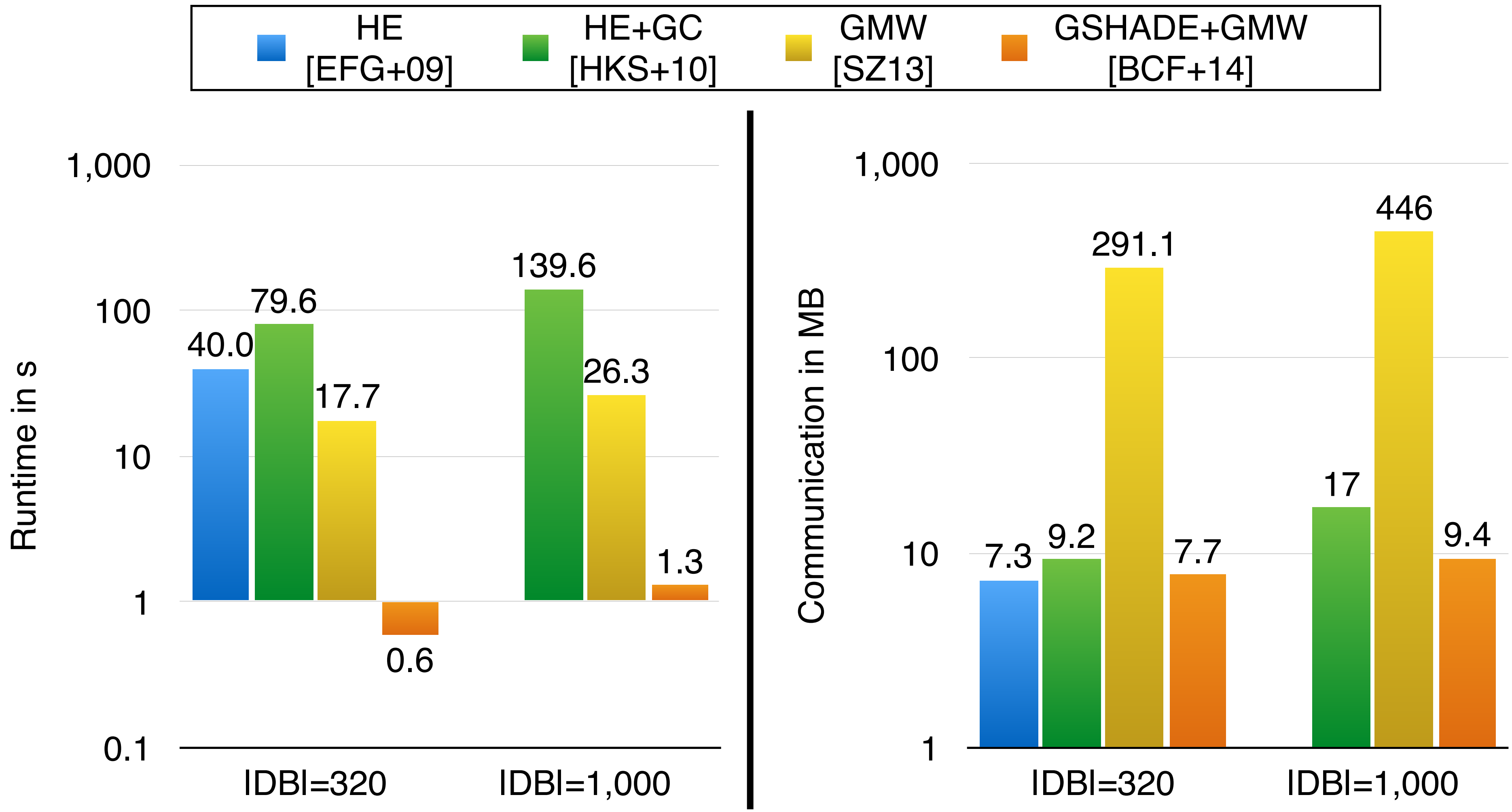
Compare biometric sample with DB of **5,000** entries.

Algorithm	Distance	Time in s	Communication in MB
SCiFI	Hamming	1.0	6.2
Eigenfaces	Euclidean	5.0	83.6
FingerCodes	Euclidean	6.7	67.5
IrisCodes	Normalized Hamming	9.1	56.4

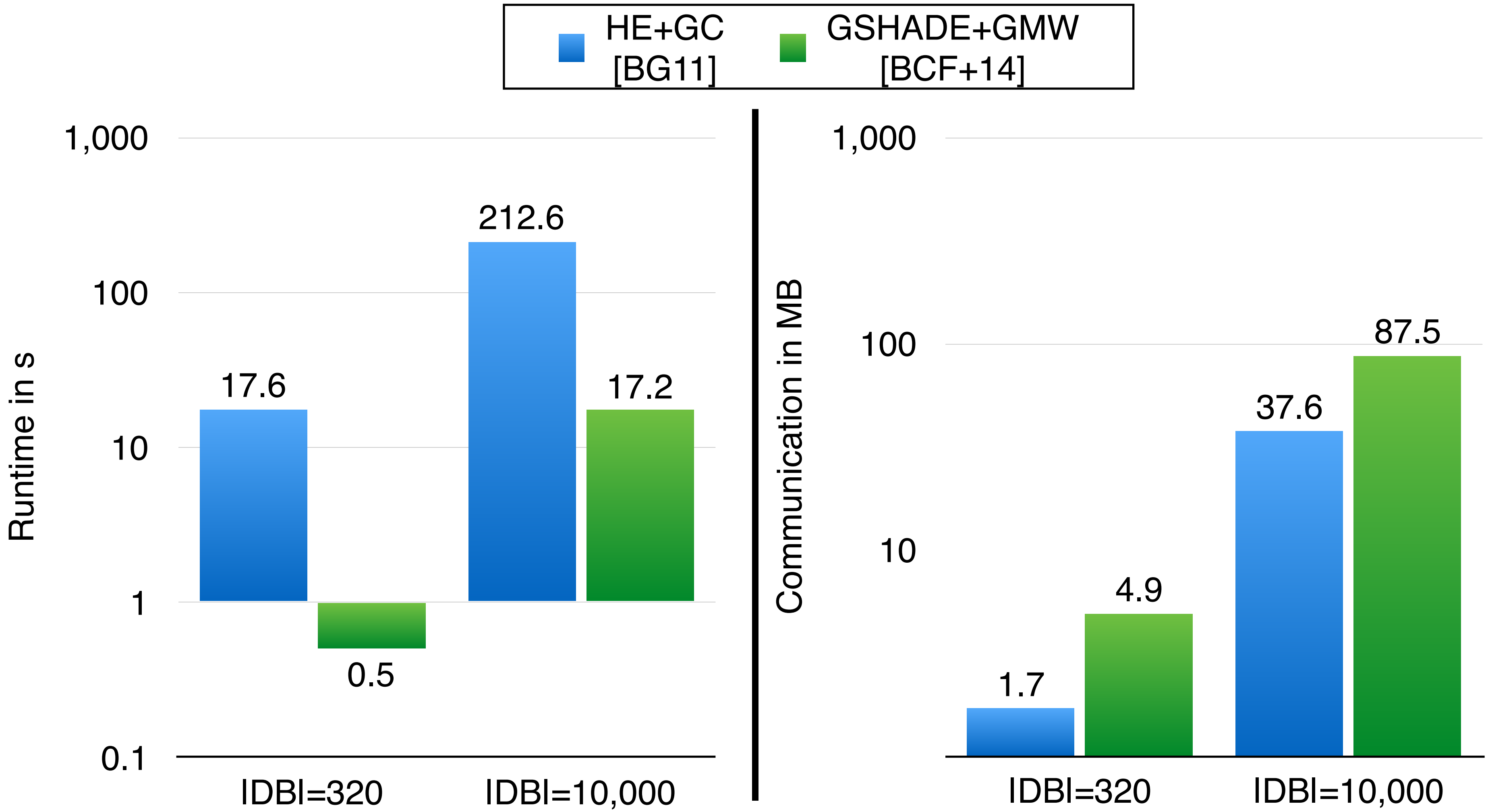
Performance for SCiFI



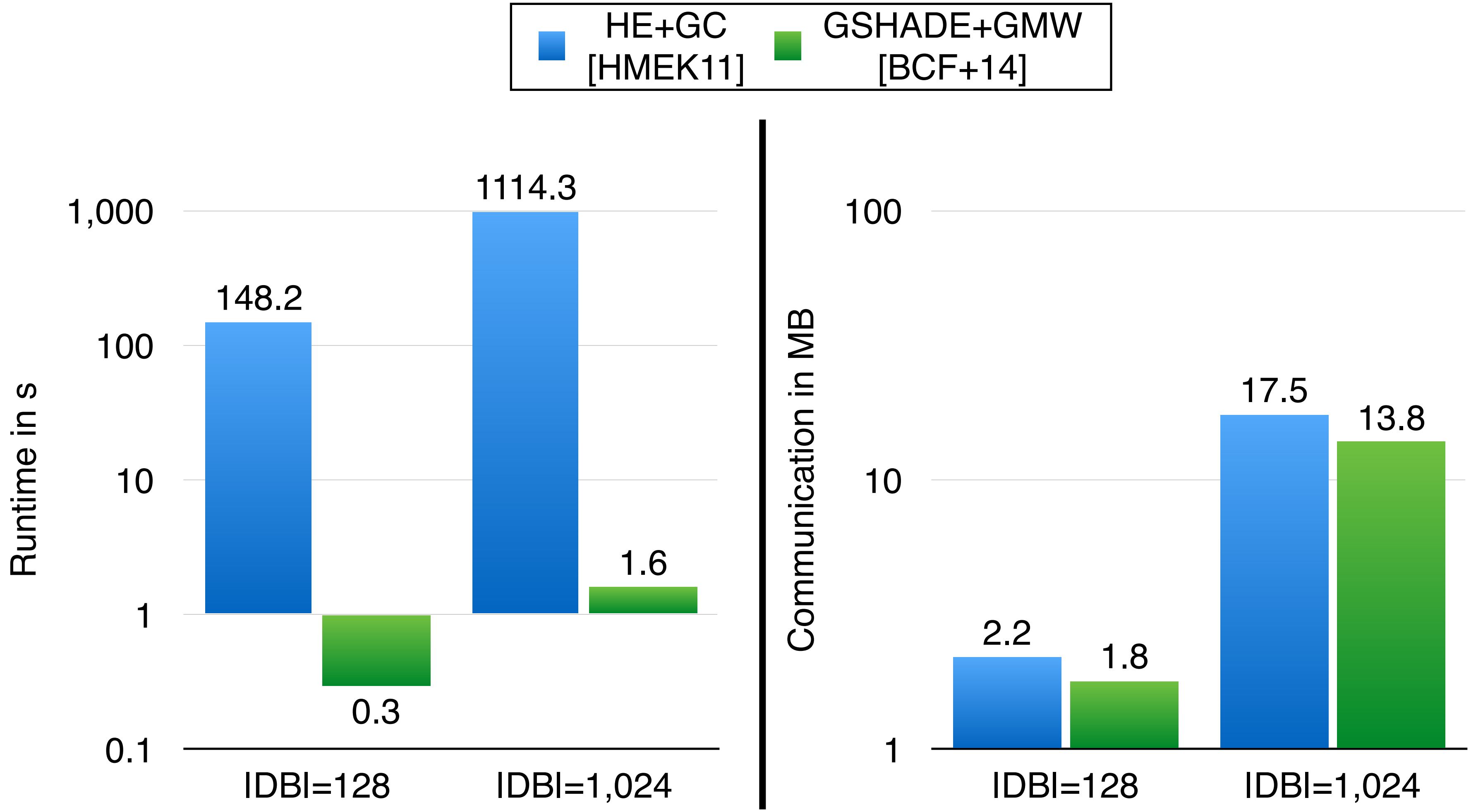
Performance for Eigenfaces



Performance for Iriscodes



Performance for Fingercodes



Summary

Conclusion

- OT is very efficient due to OT extensions
- Applications can be built efficiently directly on OT

Future Work

- Further optimize *communication* of OT / secure computation
- Other applications based directly on OT / GSHADE for other distances
- Extend to stronger adversary models

GSHADE: Faster Privacy-Preserving Distance Computation and Biometric Identification

TECHNISCHE UNIVERSITÄT DARMSTADT

Thanks for your attention.

Questions?

Contact: http://encrypto.de