A Practical Universal Circuit Construction and Secure Evaluation of Private Functions

Vladimir Kolesnikov, Bell Labs, Murray Hill, NJ, USA
http://www.cs.toronto.edu/~vlad/

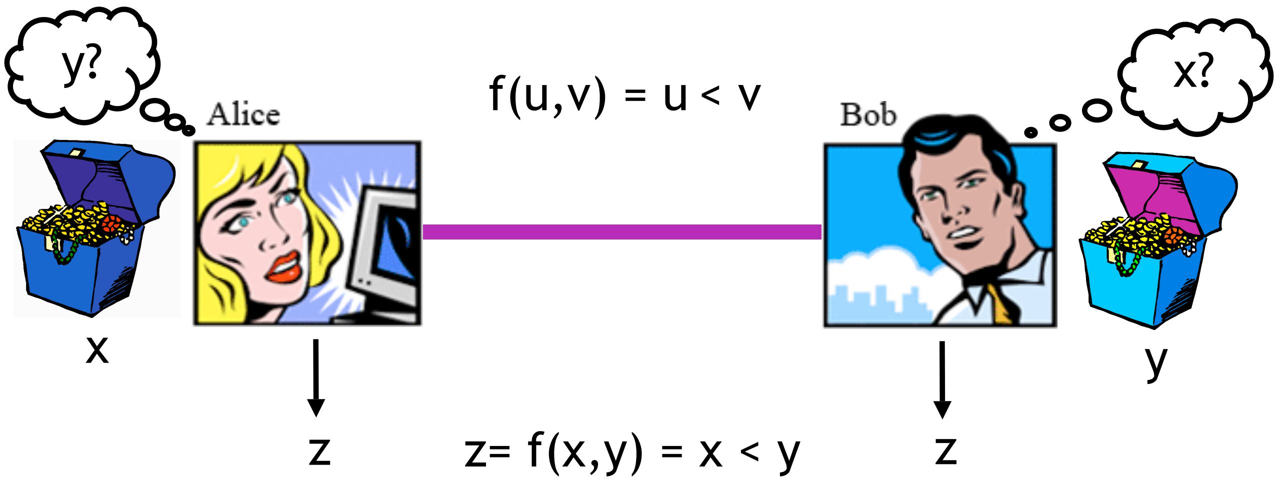
Thomas Schneider, University of Erlangen-Nuremberg, Germany
http://thomaschneider.de

Financial Cryptography and Data Security 2008, 28-31 January, Cozumel, Mexico

Agenda

1. Secure Function Evaluation (SFE)
2. Secure Function Evaluation of Private Functions (PF-SFE)
   - Reduction from PF-SFE to SFE of Universal Circuits
3. Universal Circuits (UC)
   - Valiant's UC Construction
   - A Practical UC Construction
   - Comparison w.r.t. practical PF-SFE
4. Implementation of PF-SFE

Millionaires Problem: Two millionaires want to compute who has more money while preventing the other party to learn one's private input.
They don't want to trust a third party to do the computation for them.



SFE allows two parties A and B to securely evaluate a function f(u,v) on their private inputs x and y:

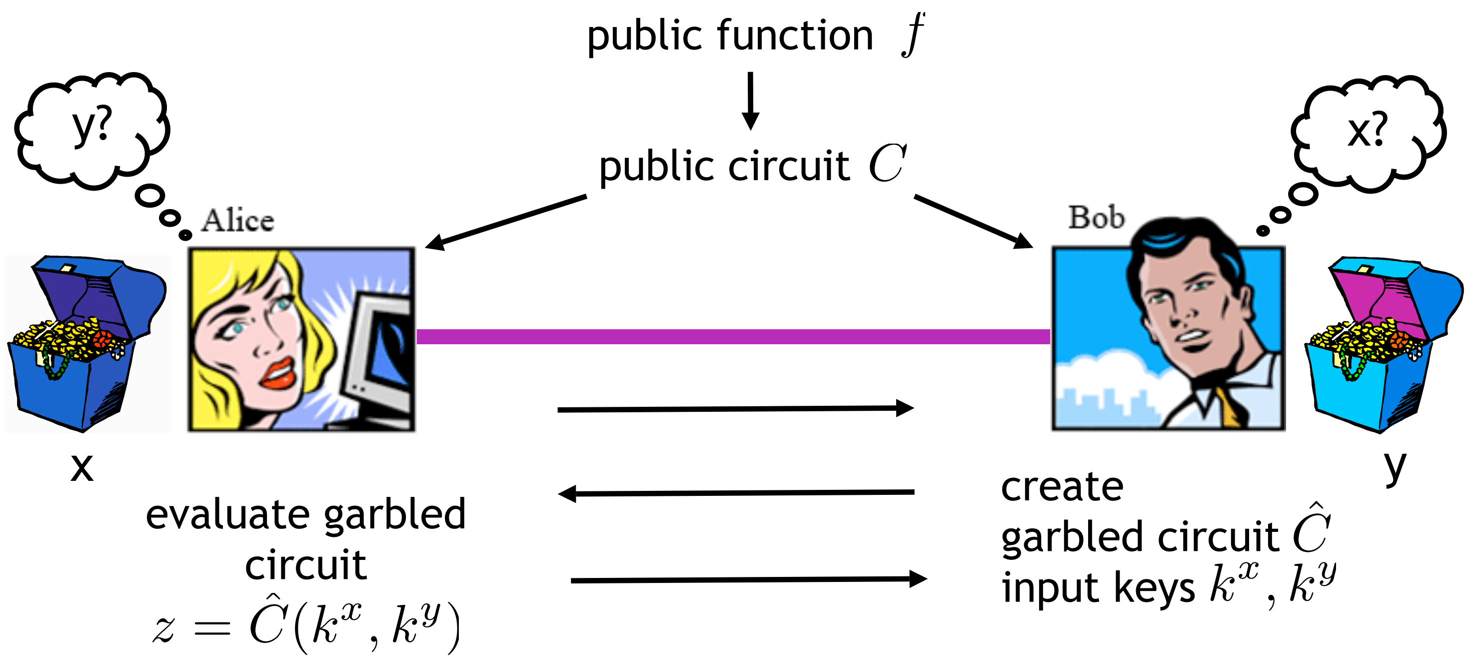
- Each party learns the result z = f(x,y).
- Each party learns nothing about the other party's secret y resp. x.

Secure Function Evaluation - One-Round Protocols

Information theoretically secure SFE protocols: [Kilian88], [IK**], [Kolesnikov05]

Computationally secure SFE protocols:

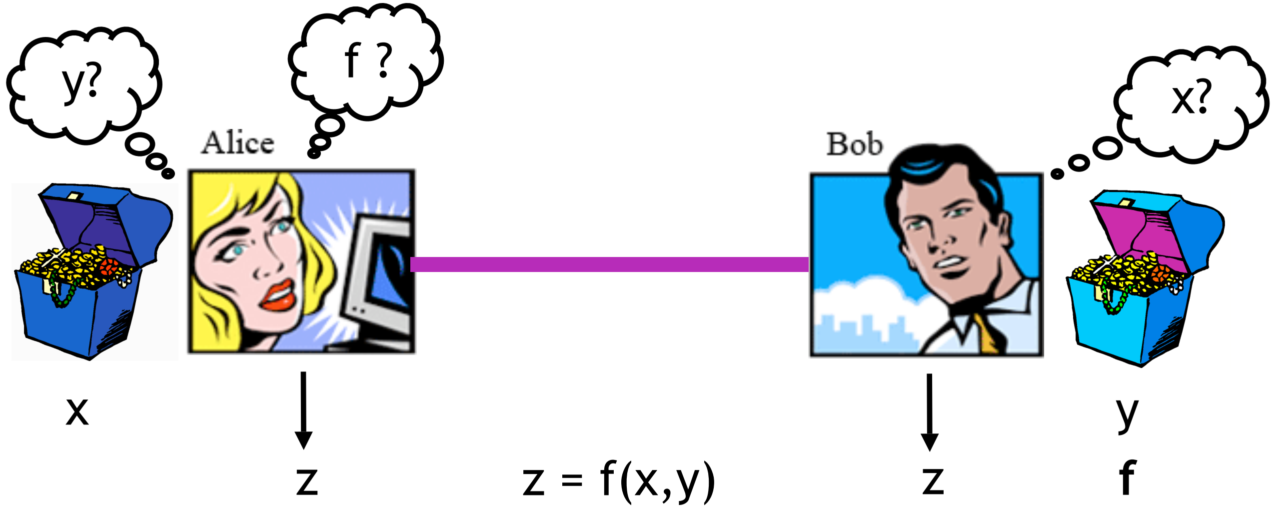
- [Yao86], [LP04] Yao's protocol
- [MNPS04] Fairplay - a secure two-party computation system



Fairplay demonstrates practicability of SFE for circuits with ~ 1M gates.

Secure Function Evaluation of Private Functions (PF-SFE)

In some applications, the function itself needs to be kept secret:



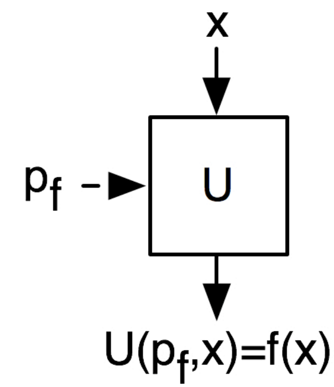
PF-SFE = SFE + everything about the function f remains secret (besides size)

Examples:

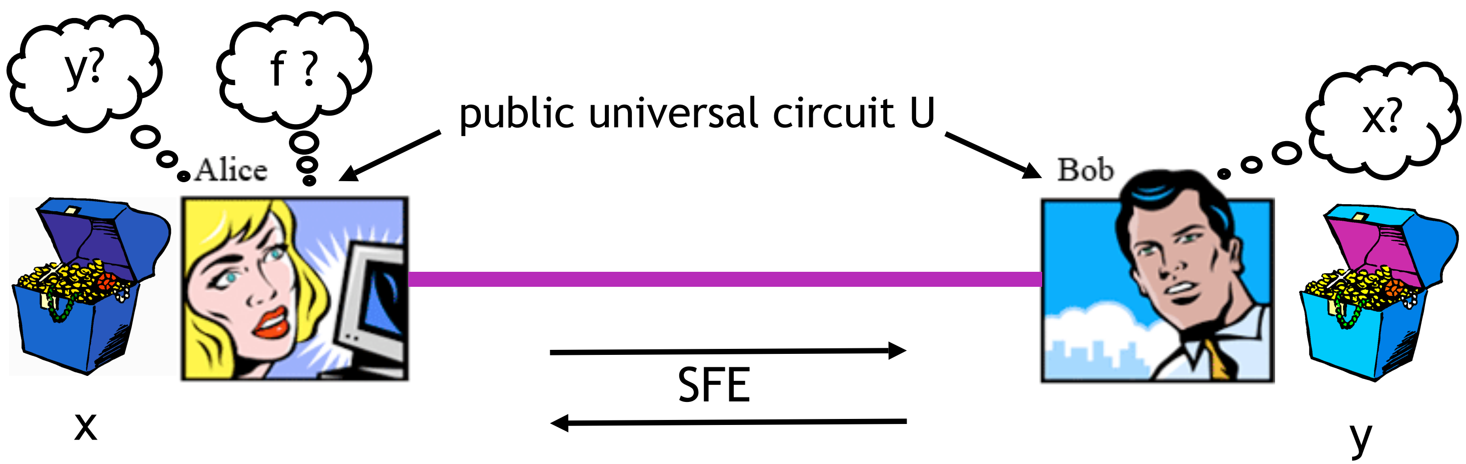
- No Fly List Checking: Bob wants to check private data x of Alice (list of passengers) with private checking function f (no-fly-function).
- Similar: Credit Report Checking, Medical History Checking
- Mobile Code: Bob executes private code f in untrustworthy environment A.
- Privacy Preserving Database Querying (e.g. patent database): Client B executes private query on private database x of DB server A.

Reduction from PF-SFE to SFE

function f ⟹ data p_f [Turing36: Universal Turing Machine]



A **universal circuit U** is a programmable circuit that takes a **function description p_f (program)** of a function f **as input** and computes the function f on the input x.



$f(x,y) = U(p_f , x, y)$

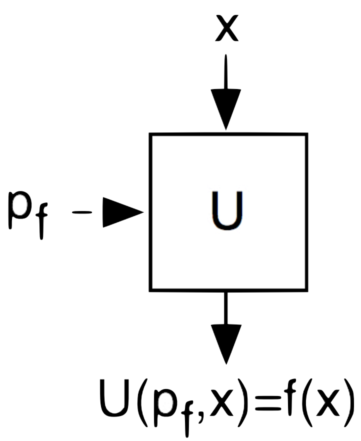
$f \Rightarrow p_f$

PF-SFE can be reduced to SFE of a universal circuit U.

Vladimir Kolesnikov
Alcatel·Lucent
Thomas Schneider
Friedrich-Alexander-Universität Erlangen-Nürnberg

Universal Circuits

A **Universal Circuit** UC_v^u is an acyclic circuit U which computes any boolean function $f : \{0,1\}^u \rightarrow \{0,1\}^v$ on input of the function description p_f (program) and the data $x \in \{0,1\}^u$.



[Valiant76: Universal Circuits]: Asymptotically optimal Construction of a ${}^kUC_v^u$ that can simulate all circuits with

- **u inputs** and **v outputs**
- **k gates** (2 inputs, 1 output, arbitrary fan-out)
- construction based on universal graphs
- no explicit programming algorithm given

$$size(UC_v) \sim (19k + 9.5u + 9.5v)\log k$$

Feasible circuit size for PF-SFE with Fairplay and Valiant's UC:

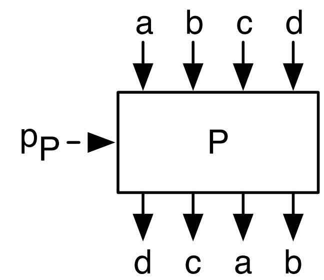
$$19k\log k < 1M \Rightarrow k < 4,354$$

⇒ Need practical UC construction for PF-SFE of circuits of that size !

Basic Building Blocks needed in our Construction

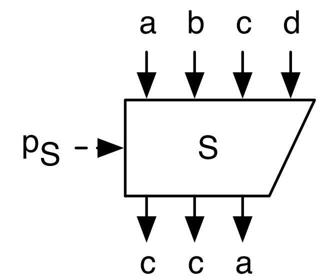
P_v^u permutation block:
Permute the u inputs to the v outputs (no duplicates)

[Waksman68: A Permutation Network]: $u = v = N = 2^n$
Efficient construction $size(P_N) = 2N \log N - 2N + 2$

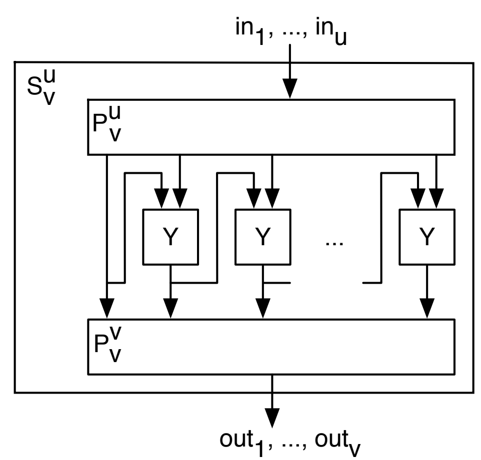


New: Generalized Permutation Blocks: arbitrary u,v

S_v^u selection block: Select for each of the v outputs
one of the u inputs (with duplicates).

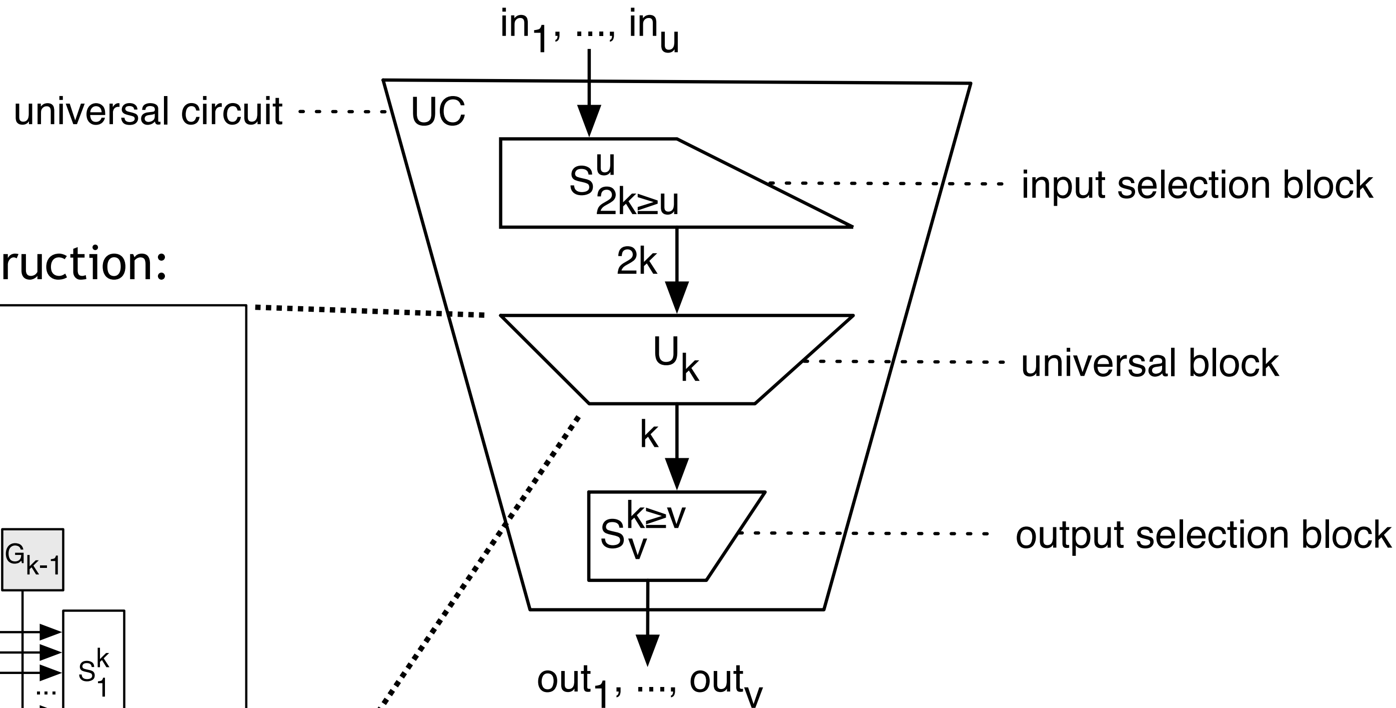


New: Permutation-based $\mathbf{S_v^u}$ Selection Blocks
size and efficient programming $\in O((u+v)\log(u+v))$

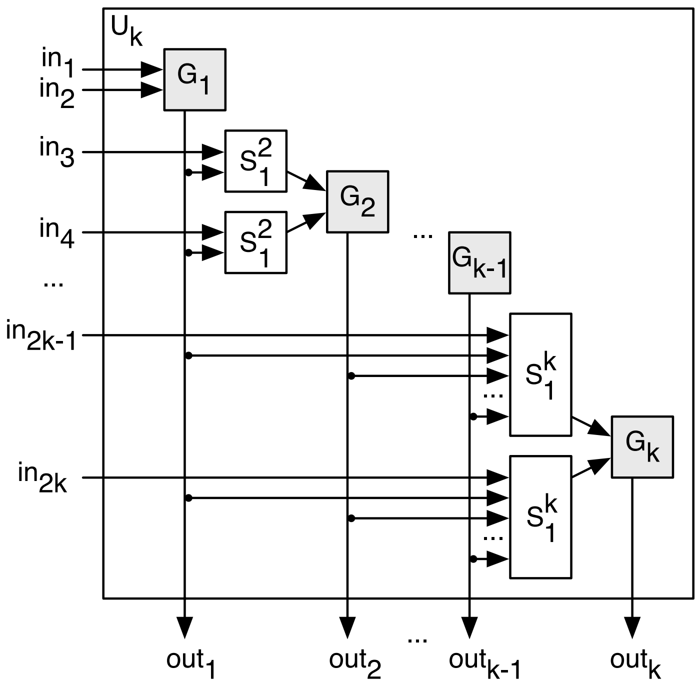


Vladimir Kolesnikov
Alcatel-Lucent
Thomas Schneider
Friedrich-Alexander-Universität Erlangen-Nürnberg

Block based Universal Circuit



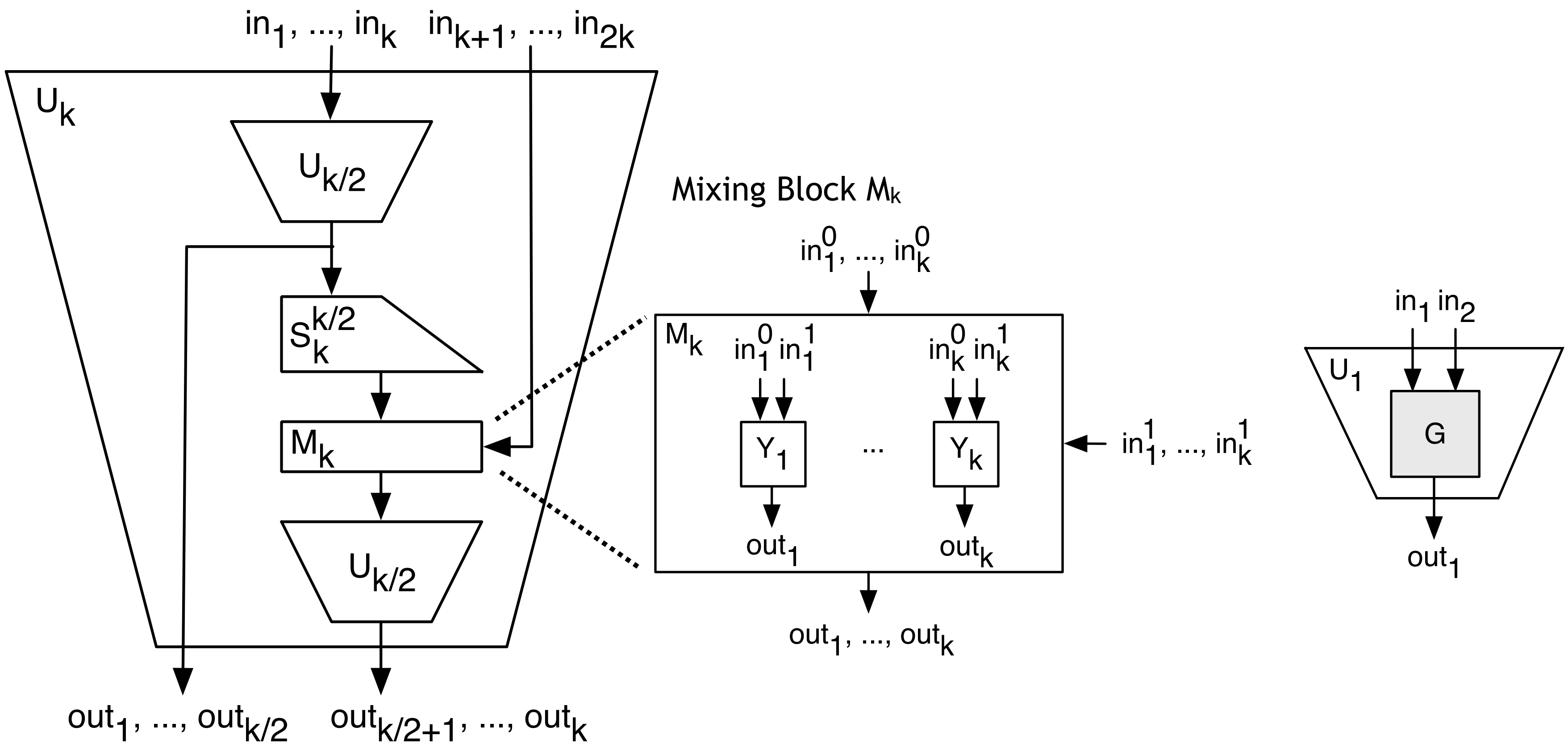
Simple U_k construction:



$$size(U_k) = k^2$$

$G_1, \ldots, G_k$ sorted topologically, i.e. G_i can have inputs from previous gates $G_{j<i}$ only.

Recursive Universal Block Construction



$$size(U_k) = 1.5k\log^2 k - 1.5k\log k + 6k - 5$$

$$size(UC_k) = 1.5k\log^2 k + 2.5k\log k + 9k + 1 + (u+2k)\log u + (k+3v)\log v - 2u - 4v$$

Comparison of our practical UC construction UC_P and Valiant's UC_V

$$size(UC_P) = 1.5k\log^2 k + 2.5k\log k + (u+2k)\log u + (k+3v)\log v + O(k)$$

$$size(UC_V) = 19k\log k + 9.5u\log k + 9.5v\log k + O(k)$$

- Asymptotically not optimal (⇒ use Valiant's UC for large circuits)
- For practical circuits our UC construction is smaller than Valiant's especially for circuits with many in- and outputs (e.g. comparison/search)
  - relative size $size_{rel} = \frac{size(UC_P)}{size(UC_V)}$, break-even point $k_{eq} = k\big|_{size_{rel}=1}$

circuit inputs and outputs	u	v	break-even point k_{eq}	relative size $size_{rel}$ $k = 1,000$	$k = 5,000$	$k = 10,000$
few	$o(k)$	$o(k)$	2,048	91.8%	110.2%	118.1%
	$0.5k$	$0.1k$	5,000	86.0%	100.1%	106.2%
	$0.5k$	$0.25k$	8,000	83.1%	96.4%	102.1%
	$1k$	$0.5k$	117,000	69.0%	79.5%	84.0%
many	$2k$	$1k$	26,663,000	53.6%	60.9%	64.1%

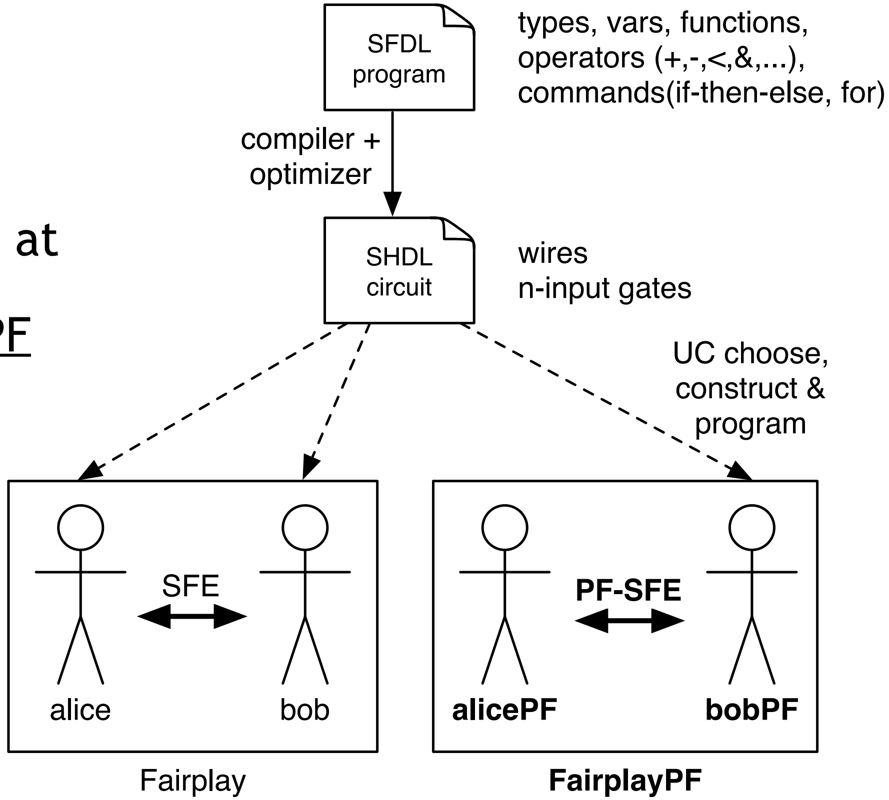
Implementation of PF-SFE based on Fairplay ⇒ FairplayPF

- Our practical recursive UC construction is small, easy and can be efficiently programmed based on the structure of the given circuit.

- Practical PF-SFE implemented as extension for Fairplay system:

  Source and Documentation available at

  http://thomaschneider.de/FairplayPF



- Coming soon:
  New SFE protocol optimized for UCs ⇒ further improve PF-SFE by factor of 4.

Thank you for your kind attention.