(EFFICIENT) ZERO-KNOWLEDGE, *(SPECIAL PURPOSE)* GARBLED CIRCUITS, *(THE SIMPLEST)* OBLIVIOUS TRANSFER,

Claudio Orlandi – Aarhus University

In this talk: 3 simple ideas from

- Jawurek, Kerschbaum, Orlandi
  - ***Zero-Knowledge from Garbled Circuits, CCS 2013***
- Frederiksen, Nielsen, Orlandi
  - ***Privacy-Free Garbled Circuits, EUROCRYPT 2015***
- Chuo, Orlandi
  - ***The Simplest OT Protocol, ePrint (next week?)***

Zero-Knowledge from Garbled Circuits

Jawurek, Ferschbaum, Orlandi

CCS 2013

Zero-Knowledge Protocols

- IP/ZK – GMR85
  - Revolutionary idea in cryptography and CS
- Important in practice
  - Authentication
  - Essential component in complex protocols
- What about efficiency?

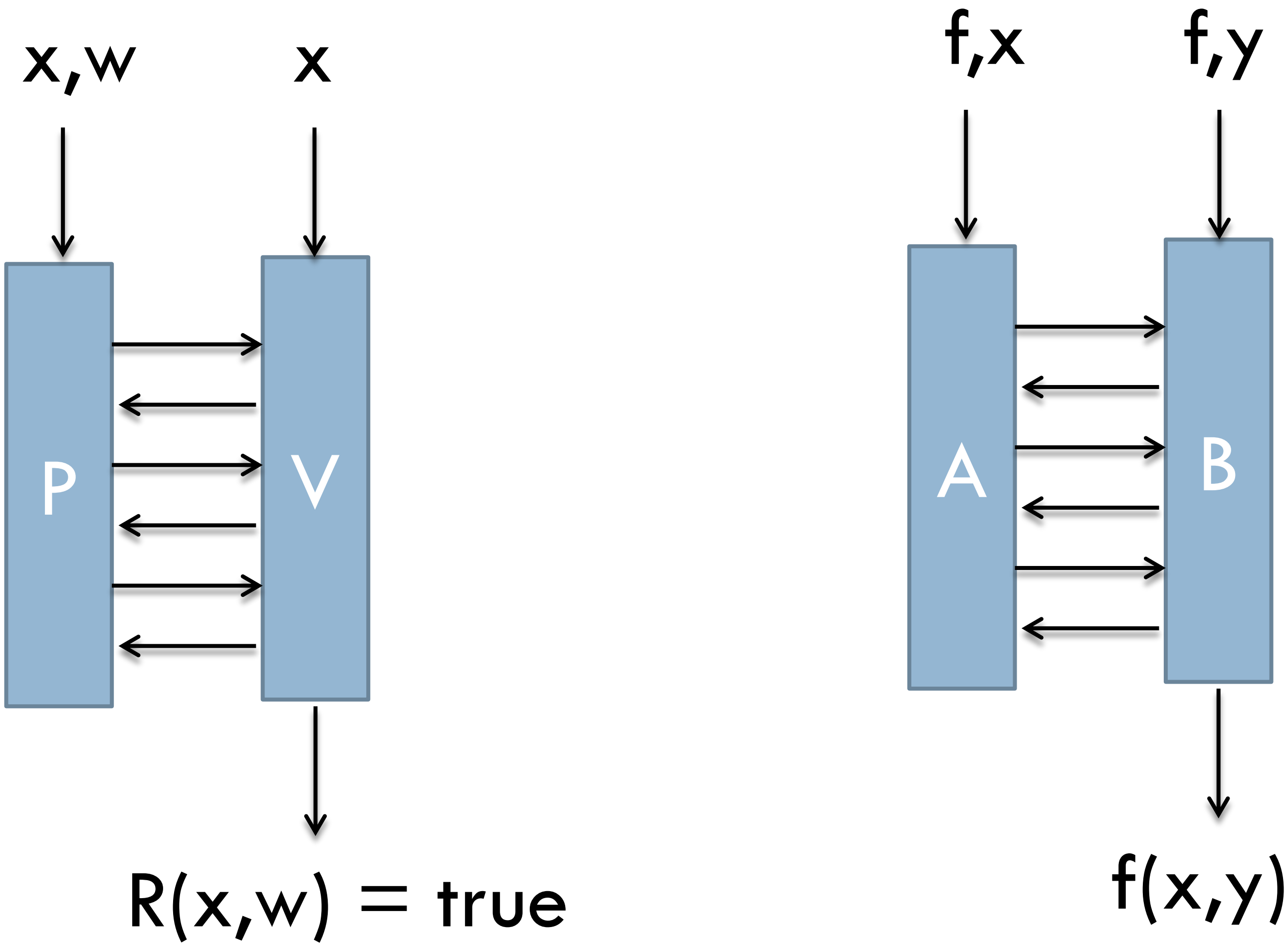
Zero-Knowledge Protocols

- Many examples of efficient ZK for algebraic languages
  - Discret Logarithm
  - RSA
  - Lattice
  - …
- What about non-algebraic statements?
  - How do I prove ”I know x s.t. y=SHA(x)”?
- This work tries to fill this gap!

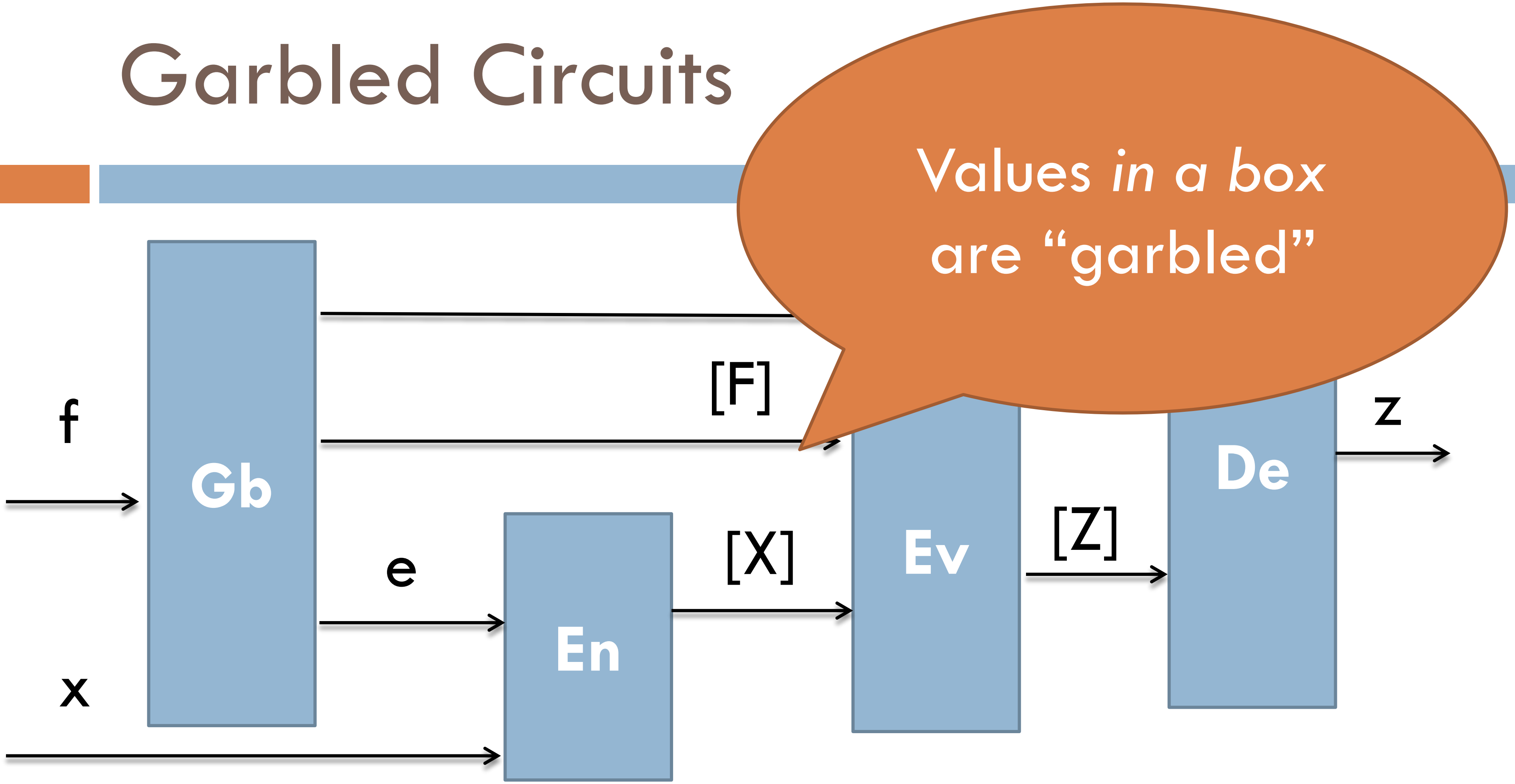
Related Work

- IKOS'07
  - ZK from (honest majority) MPC
  - First step towards the "MPC in the head" approach
- Efficient NIZK/SNARK (GOS06,GGPPR13,…)
  - Non-interactive ☺
  - Require public key operation per gate ☹

Zero-Knowledge vs Secure 2PC

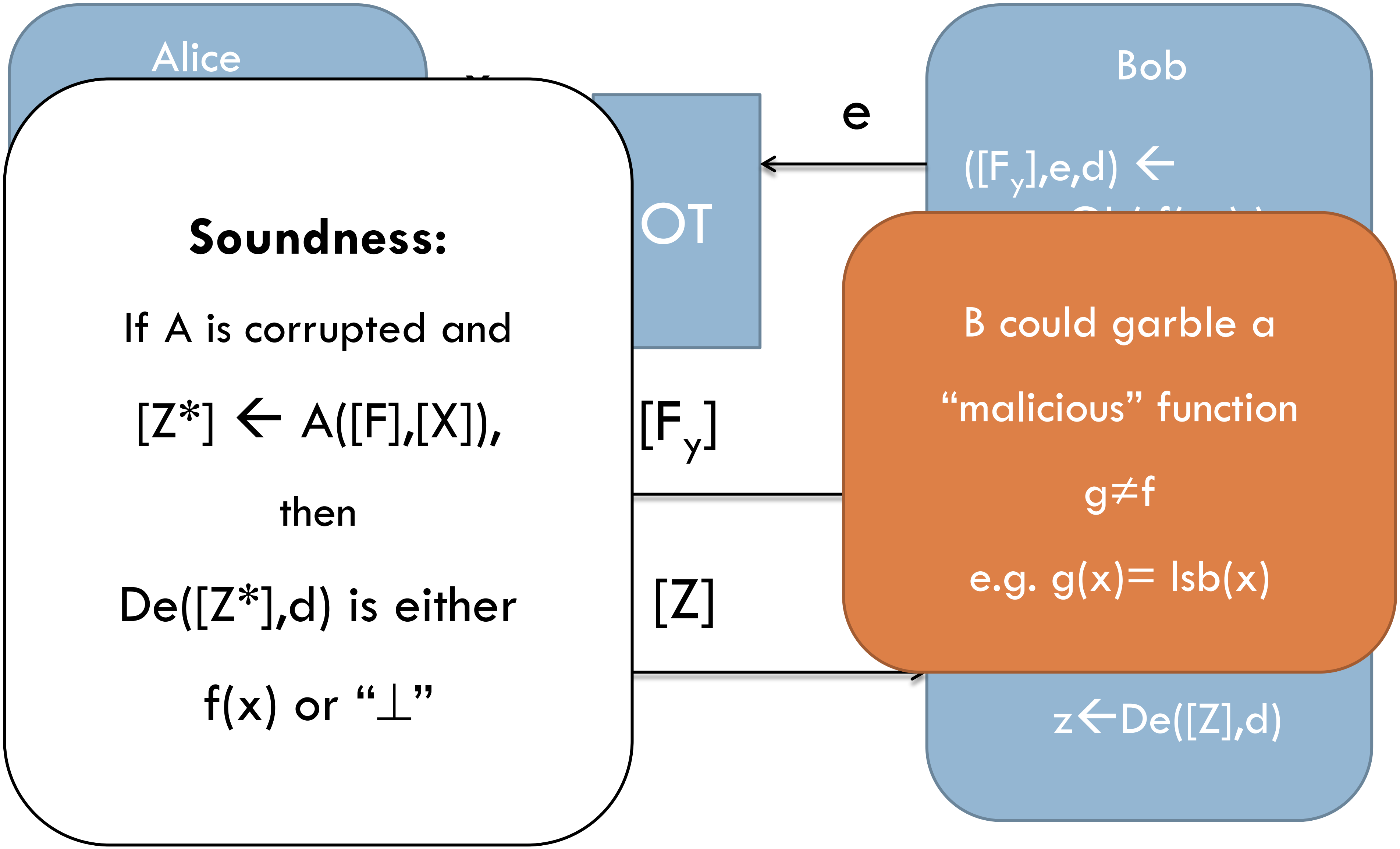


Garbled Circuits



Correct if z=f(x)

2PC from GC (Yao's protocol)



2PC secure against active adversaries?

How can Bob prove that he garbled F

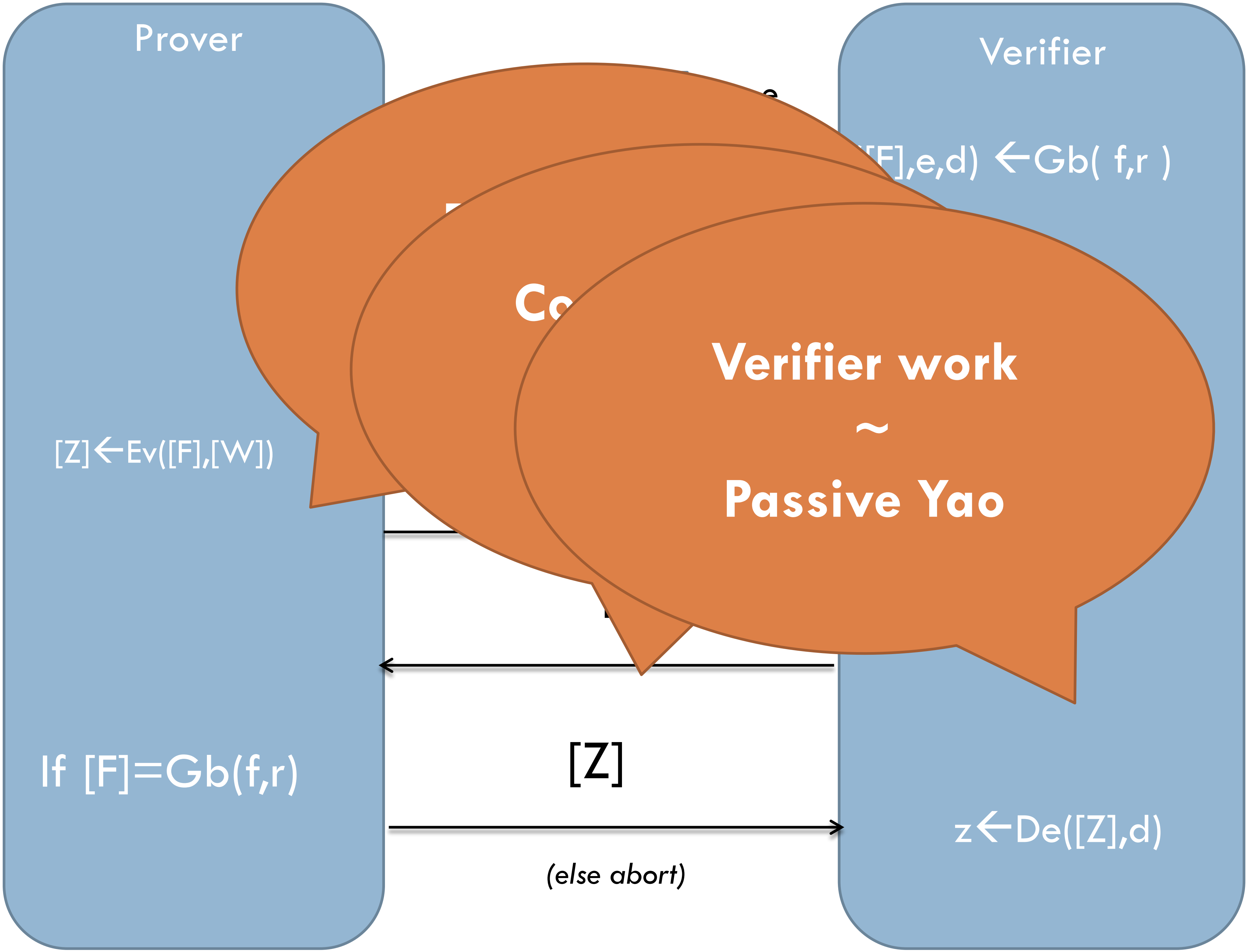
without revealing any extra information?

- Plenty of (costly) solutions are known for 2PC
  - Zero-Knowledge
  - Cut-and-choose
  - Etc.
- **Can we do better for ZK?**

ZK based on GC

- **The main idea:**
  - In ZK the verifier (Bob) has no secrets!
  - After the protocol, Bob can reveal all his randomness.
  - Alice can simply check that Bob behaved honestly

    by *redoing his entire computation*.


Prover
Verifier
[F],e,d) ←Gb(f,r)
[Z]←Ev([F],[W])
Verifier work
~
Passive Yao
If [F]=Gb(f,r)
[Z]
z←De([Z],d)
(else abort)

CCS Implementations

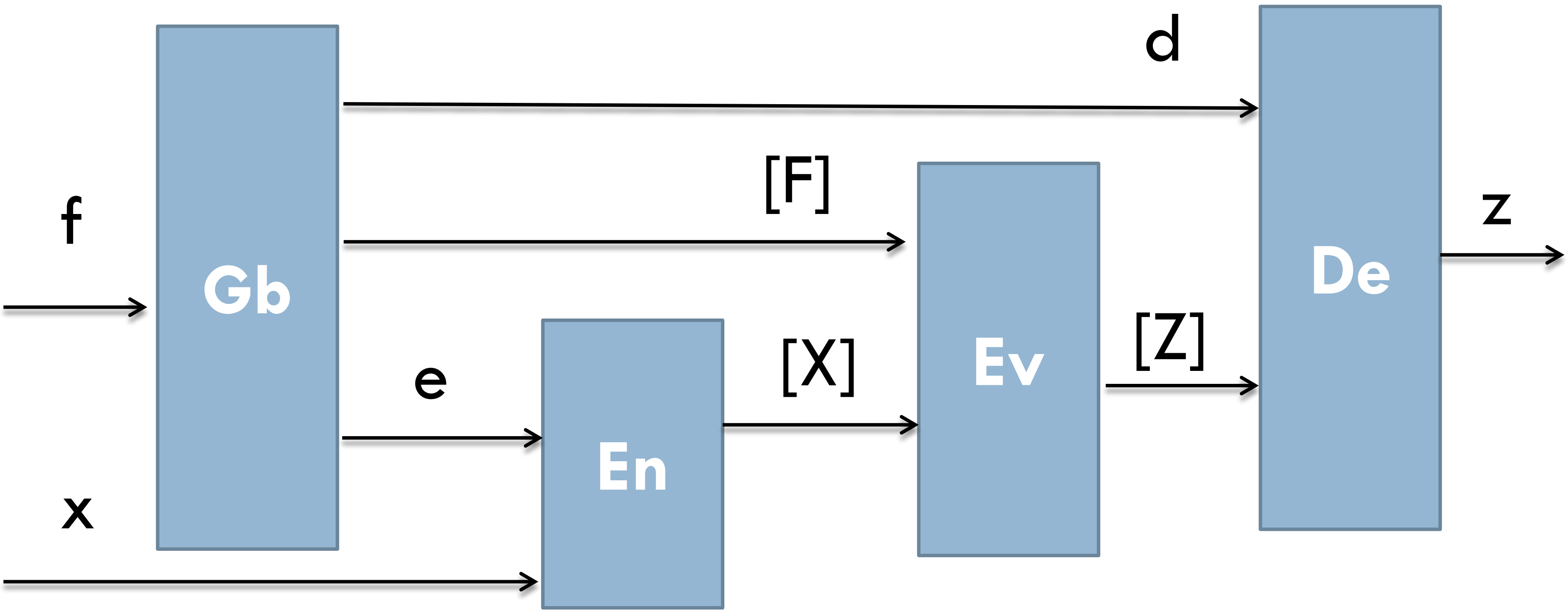
- Code not open-source, but easily reproducible
  - FastGC garbled circuits implementation
  - Smart-Tillich optimized circuits: AES, MD5, SHA…
  - GCParser to combine the two above
  - SCAPI for implementing OT (using elliptic curves)

Privacy-Free Garbled Circuits

Frederiksen, Nielsen, Orlandi

EUROCRYPT 2015

Garbled Circuits



Correct if z=f(x)

Main idea

- In 2PC GC ensure that evaluator does not learn internal values
  - In Yao garbled circuits evaluation must be oblivious
- But in ZK the prover knows all the input bits!
  - He also knows all internal wires values
- Can we optimize?
  - Yes!

Garbling Schemes without Privacy

- **Conceptual contribution:**
  - Natural separation between **privacy** and **authenticity**
- **Concrete efficiency:**
  - Better constants in garbled circuit

Can we construct garbling schemes tailored to specific applications, which are more efficient than Yao's original construction?

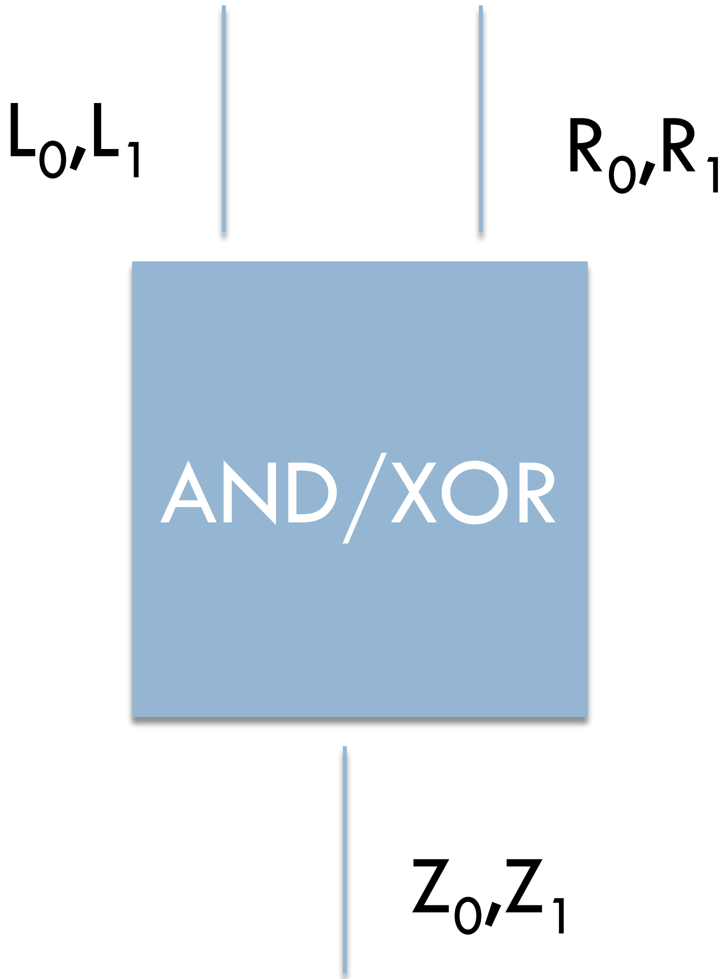
Performances for m-ary gate

		Garbler H/gate	Eval H/gate	Communication bit/gate
GRR1	AND	m+1	1	k(m-1)
	XOR	-	-	k(m-1)
Free-XOR	AND	m+1	1	km
	XOR	-	-	-

Communication (amortized # of ciphertexts per gate)									
Circuit	# of Gates		Private			**Privacy-free**			Saving
	AND	XOR	GRR2	free-XOR	fleXOR	**GRR1**	**free-XOR**	**fleXOR**	
DES	18124	1340	2.0	2.79	1.89	1.0	1.86	**0.96**	49%
AES	6800	25124	2.0	0.64	0.72	1.0	**0.43**	0.51	33%
SHA-1	37300	24166	2.0	1.82	1.39	1.0	1.21	**0.78**	44%
SHA-256	90825	42029	2.0	2.05	1.56	1.0	1.37	**0.87**	44%

Computation (amortized # of encryptions per gate for garbler/evaluator)							
Circuit	# of Gates		Private			**Privacy-free**	Saving
	AND	XOR	GRR2	free-XOR	fleXOR	**GRR1/free-XOR/fleXOR**	
DES	18124	1340	4.0/1.0	3.72/0.93	3.78/0.96	**2.79/0.93**	25%/0%
AES	6800	25124	4.0/1.0	0.85/0.21	1.44/0.51	**0.64/0.21**	25%/0%
SHA-1	37300	24166	4.0/1.0	2.43/0.61	2.78/0.78	**1.82/0.61**	25%/0%
SHA-256	90825	42029	4.0/1.0	2.73/0.68	3.11/0.87	**2.05/0.68**	25%/0%

Notation



- A *(privacy-free)* garbled gate is a gadget that given two inputs keys gives you the right output key *(and nothing else)*

- $(Z_0, Z_1, gg) \leftarrow Gb(L_0, L_1, R_0, R_1)$
- $Z_{g(a,b)} \leftarrow Ev(L_a, R_b, gg)$

Garbling w/o free-XOR (GRR1)

Gb_AND(L_0, L_1, R_0, R_1)

- Output keys:
  - $Z_1 = H(L_1, R_1)$
  - $Z_0 = H(L_0)$
- Send:
  - $C = Z_0 \oplus H(R_0)$

Ev_AND(L_x, R_y, C)

- If($x = y = 1$)

  output $Z_1 = H(L_x, R_y)$
- If($x = 0$)

  output $Z_0 = H(L_x)$
- If($y = 0$)

  output $Z_0 = C \oplus H(R_y)$

Garbling w/o free-XOR (GRR1)

Gb_XOR(L_0,L_1,R_0,R_1)

- Output keys:
  - $Z_0 = L_0 \oplus R_0$
  - $Z_1 = L_1 \oplus R_0$
- Send:
  - $C = L_0 \oplus R_0 \oplus L_1 \oplus R_1$

Ev_XOR(L_a, R_b, C)

- If(a = 0) output
  $Z_{(a \oplus b)} = L_a \oplus R_b$
- If(a = 1) output
  $Z_{(a \oplus b)} = C \oplus L_a \oplus R_b$

Conclusions & Open Problems

- Still a lot to be done with garbling schemes!
- Other specific purpose garbling schemes?
- Non-interactive ZK (w/o PKE/gate)?

The Simplest Oblivious Transfer Protocol

Chou, Orlandi

coming soon on ePrint

Diffie Hellman Key Exchange

m

$X = g^x$ →

← $Y = g^y$

$K = H(Y^x)$

$K = H(X^y)$

There is another key $K' = H((X/Y)^x)$ which Bob cannot compute!

$C = E(K,m)$ →

$m = D(K,C)$

The Simplest OT protocol

m_0, m_1

b

$X = g^x$ →

$b=0 : Y = g^y$

$b=1 : Y = X/g^y$

← Y

$K_0 = H(Y^x)$

$K_1 = H((X/Y)^x)$

$K_b = H(X^y)$

$C_0 = E(K_0, m_0)$

$C_1 = E(K_1, m_1)$ →

$E((\alpha,\beta), m) =$

$(\alpha + m, (\alpha + m)\beta)$

$m_b = D(K_b, C_b)$

The Simplest OT Protocol

- Complexity:
  - Communication: 1ge/OT + 2 ctxt/OT + 1ge
  - Computation: 3 exp/OT + 3 H/OT + 2 exp
- Security:
  - UC vs. active adversary with programmable RO
- Performances: ~0.2ms/OT @ 64 OTs
  - Implementation based on Bernstein's Curve25519