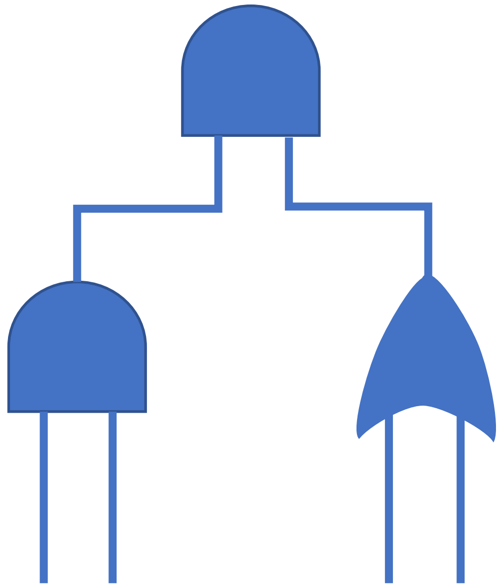
Better Concrete Security for Half-Gates Garbling (in the Multi-Instance Setting)

Chun Guo Jonathan Katz Xiao Wang
Chenkai Weng Yu Yu

Yao's garbled circuits

- Two-party computation (2PC)
- Multiple optimizations
  - Point-and-permute
  - Free-XOR
  - Garbled-row-reduction
  - **Half-gates (state-of-the-art) [1]**
  - **Fixed-key AES based garbling [2]**



[1] S. Zahur, M. Rosulek, and D. Evans. Two halves make a whole—reducing data transfer in garbled circuits using half gates. In Advances in Cryptology—Eurocrypt 2015, Part II, volume 9057 of LNCS, pages 220–250. Springer, 2015.
[2] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient garbling from a fixed-key blockcipher. In IEEE Symposium on Security and Privacy (S&P) 2013, pages 478–492, 2013.

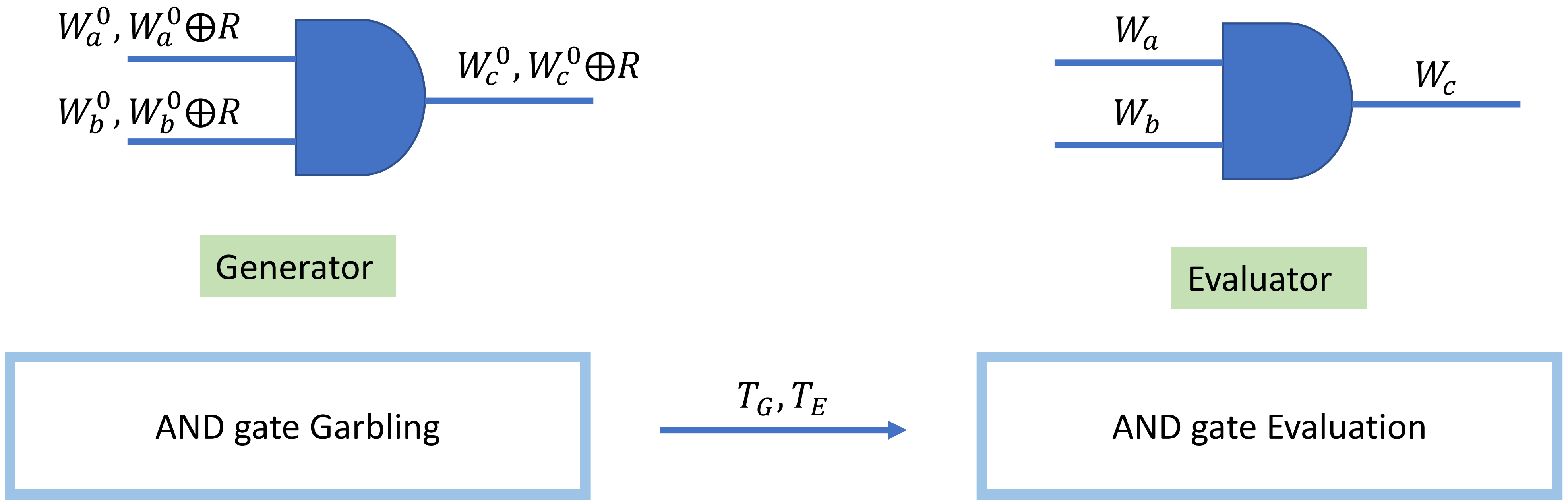
Concrete security for Half-Gates (Outline)

- An attack on current Half-Gates implementation
- Deficiencies of current implementation
  - Inappropriate instantiation of the hash function
  - A lack of concrete security
- A new abstraction of hash function
  - miTCCR hash
  - Better concrete security
  - Optimization/performance

Attack overview

- Exploit the weakness when $H(*)$ instantiated with fixed-key AES
- Attacker succeed in running time $O\left(2^k/C\right)$
  - k: bit length of the labels; C: # of AND gates
  - Circuit with $k = 80$ and $C = 2^{40}$ would be completely broken
  - Circuit with $k = 128$ and $C = 2^{40}$ has only ~80 bit security
- Implementation of the attack consistent with analysis
- Can be extended to multi-instance case

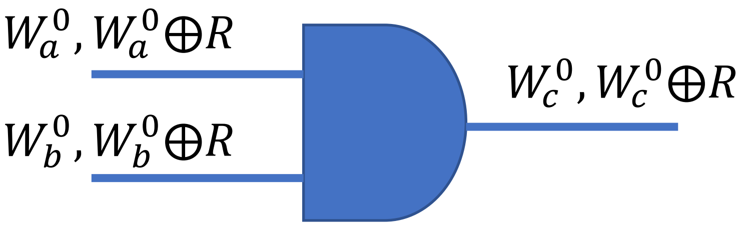
Half-gate protocol

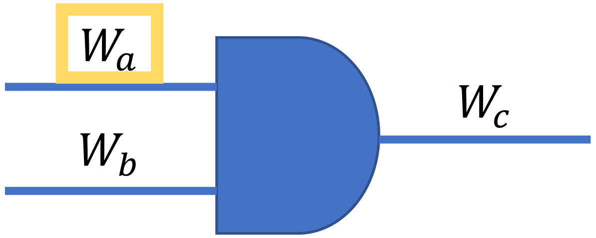


$$T_G = H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R$$

$$T_E = H\left(W_b^0, j'\right) \oplus H\left(W_b^1, j'\right) \oplus W_a^0$$

Half-gate protocol





Generator

Evaluator

AND gate Garbling

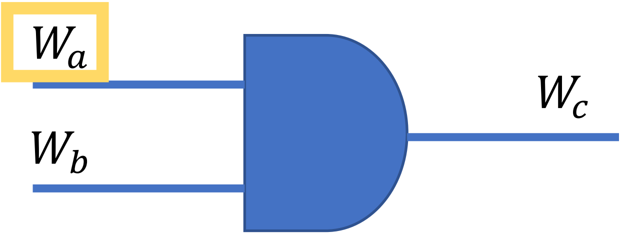
T_G, T_E →

AND gate Evaluation

$$T_G = H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R$$

$$T_E = H\left(W_b^0, j'\right) \oplus H\left(W_b^1, j'\right) \oplus W_a^0$$

Details of the attack



Evaluator

- The evaluator receives $T_G = H(W_a^0, j) \oplus H(W_a^1, j) \oplus p_b R$
- Compute

$$H_a \stackrel{\text{def}}{=} T_G \oplus H(W_a, j) = H(W_a \oplus R, j) \oplus p_b R$$

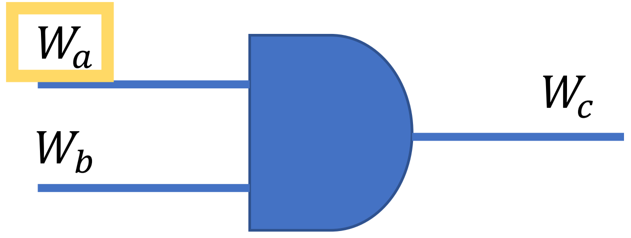
- With prob=1/2,

$$H_a = H(W_a \oplus R, j)$$

Details of the attack

Implementation of the H :

$H(x, j) = \pi(K) \oplus K$, where $K = 2x \oplus j$



Evaluator

- With prob=1/2,

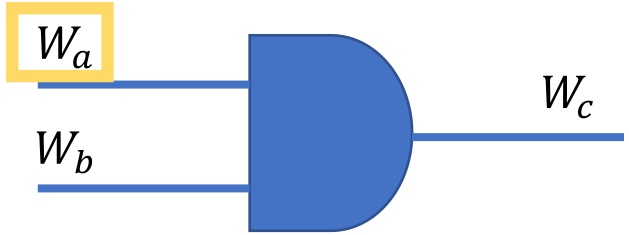
$$H_a = H(W_a \oplus R, j) = \pi(2(W_a \oplus R) \oplus j) \oplus 2(W_a \oplus R) \oplus j$$

- If find W^* s.t. $H_a = \pi(W^*) \oplus W^*$, then knows R.
- The evaluator collects all the (j, W_a, H_a) pairs.

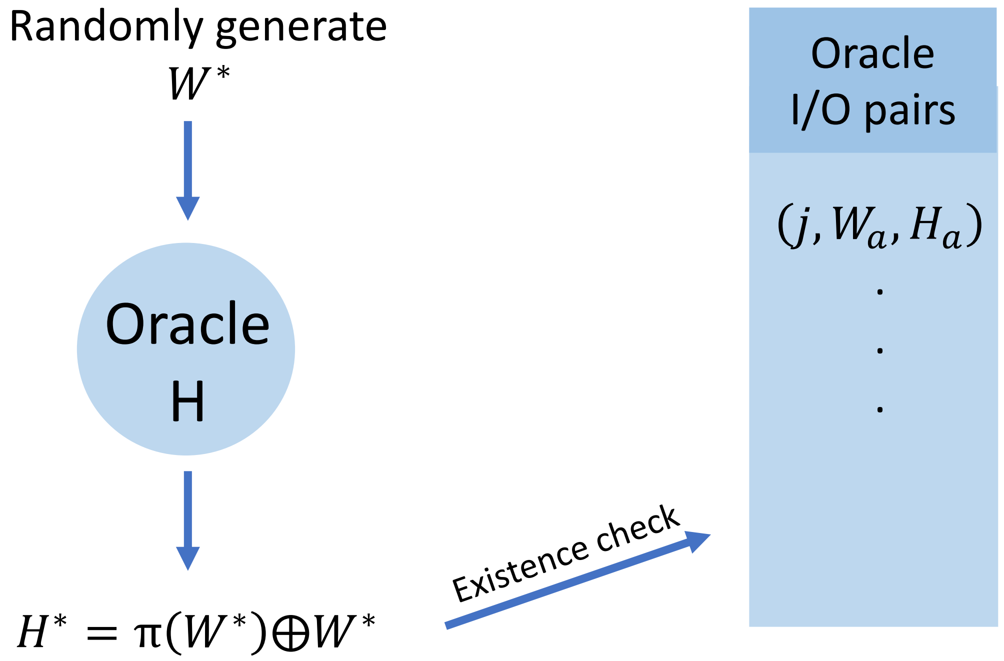
Details of the attack

Implementation of the H :

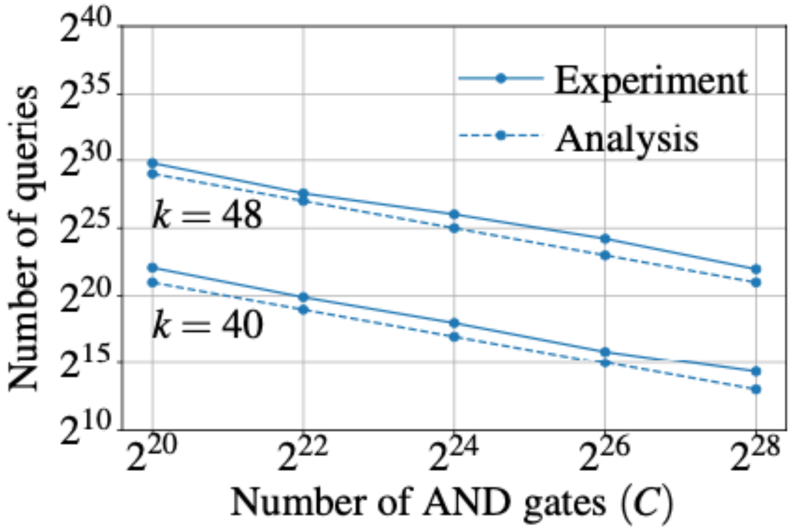
$H(x, j) = \pi(K) \oplus K$, where $K = 2x \oplus j$



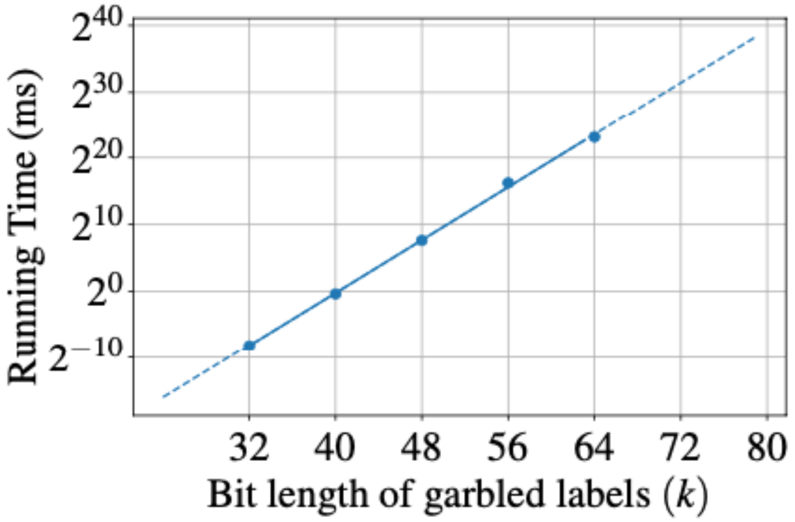
Evaluator



Implementation of the attack



(a) Number of π-queries for the attack to succeed, on a log/log scale.



(b) The running time of our attack with $C = 2^{30}$ and different values of k.

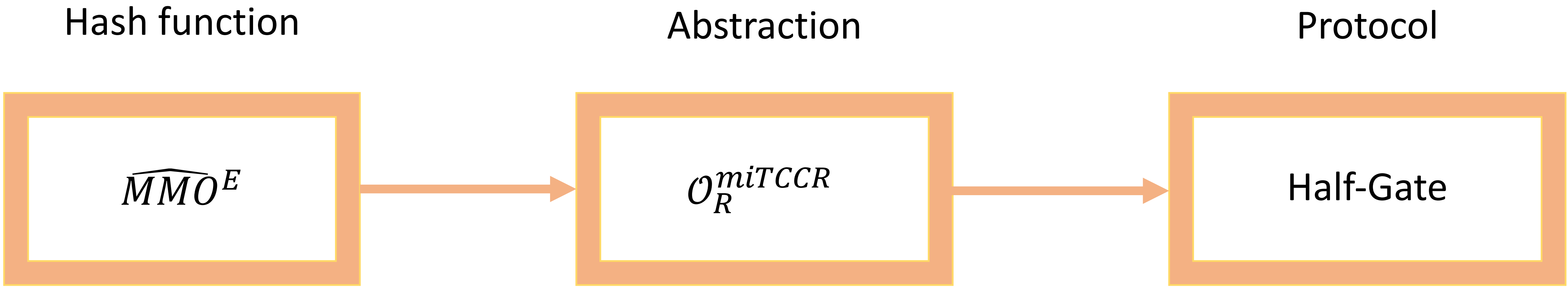
Result of interpolation:
Breaking the circuit when k=80 using 267 machine-months & $3500.

Better concrete security

Abstraction

$\mathcal{O}_R^{miTCCR}$

Better concrete security



Abstraction of the hash function

$$\mathcal{O}_R^{miTCCR}(w, i, b) \stackrel{\text{def}}{=} H(w \oplus R, i) \oplus b \cdot R$$

Definition 3. *Given a function $H^E : \mathcal{W} \times \mathcal{T} \to \mathcal{W}$, a distribution $\mathcal{R}$ on $\mathcal{W}$, and a distinguisher D, define*

$$\mathbf{Adv}_{H,\mathcal{R}}^{\mathsf{miTCCR}}(D, u, \mu) \stackrel{\text{def}}{=} \left| \Pr_{R_1,\ldots,R_u \leftarrow \mathcal{R}} \left[D^{E, \mathcal{O}_{R_1}^{\mathsf{miTCCR}}(\cdot), \ldots, \mathcal{O}_{R_u}^{\mathsf{miTCCR}}(\cdot)} = 1 \right] - \Pr_{f_1,\ldots,f_u \leftarrow \mathsf{Func}_{\mathcal{W} \times \mathcal{T} \times \{0,1\}, \mathcal{W}}} \left[D^{E, f_1(\cdot), \ldots, f_u(\cdot)} = 1 \right] \right|,$$

where both probabilities are also over choice of E and we require that

- Adversary given u instances
- Queries of form $(\star, i, \star)$ at most μ

The hash function

- Hash function (from ideal cipher)

$$\widehat{MMO}^E(x, i) \stackrel{\text{def}}{=} E\big(i, \sigma(x)\big) \oplus \sigma(x)$$

- $\sigma(x)$ is a linear orthomorphism
  - Linear if $\sigma(x \oplus y) = \sigma(x) \oplus \sigma(y)$
  - Orthomorphism if it is a permutation, and $\sigma'(x) \stackrel{\text{def}}{=} \sigma(x) \oplus x$ is also a permutation
  - $\sigma(x_L \parallel x_R) = x_R \oplus x_L \parallel x_L$
- E is modeled as an ideal cipher

Concrete security bound

- Multi-instance tweakable circular correlation robustness (miTCCR)

$$\mathcal{O}_R^{miTCCR}(w, i, b) \stackrel{\text{def}}{=} H(w \oplus R, i) \oplus b \cdot R$$

  - Adversary given u instances.
  - Queries of form $(\star, i, \star)$ at most μ.
- Attacker advantage

$$\varepsilon = \frac{2\mu p}{2^\rho} + \frac{(\mu - 1)q}{2^\rho}$$

Better concrete security for multi-instance

- Multi-instance tweakable circular correlation robustness (miTCCR)

$$\mathcal{O}_R^{miTCCR}(w, i, b) \stackrel{\text{def}}{=} H(w \oplus R, i) \oplus b \cdot R$$

- Bound the queries of form $(\star, i, \star)$.
  - Before: i starts from 1.
  - Now: i starts from a random point.
  - Proof using "balls-and-bins"

Better concrete security for multi-instance

- Multi-instance tweakable circular correlation robustness (miTCCR)

$$\mathcal{O}_R^{miTCCR}(w, i, b) \stackrel{\text{def}}{=} H(w \oplus R, i) \oplus b \cdot R$$

- Concrete security

$$\varepsilon = \frac{\mu p + (\mu - 1)C}{2^{k-2}} + \frac{(2C)^{\mu+1}}{(\mu+1)! \times 2^{\mu L}}$$

Better concrete security for multi-instance

- Multi-instance tweakable circular correlation robustness (miTCCR)

$$\mathcal{O}_R^{miTCCR}(w, i, b) \stackrel{\text{def}}{=} H(w \oplus R, i) \oplus b \cdot R$$

- Concrete security

k (bit)	C	Comp. sec. (bit)	Sta. sec. (bit)
80	$\leq 2^{43.5}$	78	40
128	$\leq 2^{61}$	125	64

Implementation & optimization

$$\widehat{MMO}^E(x,i) \stackrel{\text{def}}{=} E(i,\sigma(x)) \oplus \sigma(x)$$

- Linear orthomorphism
  - mask = _mm_set_epi64x(1^{64},0^{64})
  - $\sigma(x) = \text{\_mm\_shuffle\_epi32}(\text{a}, 78) \oplus \_mm\_and\_si128(a, \text{mask})$
- Batch key scheduling [GLNP15]
  - Batch 8 key expansion

Hash function	NI support?	k	Comp. sec. (bits)	100 Mbps	2 Gbps	localhost
Zahur et al.	Y	128	89	0.4	7.8	23
SHA-3	N	128	125	0.27	0.27	0.28
SHA-256	N	128	125	0.4	1.1	1.2
SHA-256	Y	128	125	0.4	2.1	2.45
$\widehat{\text{MMO}}^E$	Y	128	125	0.4	7.8	15
$\widehat{\text{MMO}}^E$	Y	88	86	0.63	12	15

We optimized it to 20 since then

Implementation & optimization

$$\widehat{MMO}^E(x,i) \stackrel{\text{def}}{=} E\big(i,\sigma(x)\big)\oplus\sigma(x)$$

- Linear orthomorphism
- Batch key scheduling [GLNP15]
- Implementation in EMP-toolkit
  - https://github.com/emp-toolkit/emp-tool/blob/release-2/emp-tool/utils/mitccrh.h
- Full version of the paper
  - https://eprint.iacr.org/2019/1168.pdf

EMP

Efficient Multi-Party computation toolkit