Can LWC and PEC be Friends?:
Evaluating Lightweight Ciphers in Privacy-enhancing Cryptography

Kalikinkar Mandal[1] and Guang Gong[2]

[1]Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3

[2]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, N2L 3G1, CANADA

October 20, 2020

NIST LWC Workshop 2020, Gaithersburg, USA

Outline

1. Applications
2. Binary Circuits of LWC Candidates
3. Secure Evaluation
   - 2PC computation
   - Homomorphic evaluation
4. Summary

Lightweight AE and Hash

Protecting communication: security and privacy of data in transit

- Confidentiality, Authenticity, Integrity

Storage security: encrypted data at rest

- Data security

Enabling computation over encrypted data: Privacy-enhancing cryptographic (PEC) techniques

- Secure multiparty computation (SMPC)
- Fully homomorphic encryption (FHE)
- Zero-knowledge Proofs (ZKPs)

Friendly interface between lightweight ciphers and PEC techniques

Analytics Outsourcing using FHE[1]

Thermostat

Cloud

Trusted setup

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{HKeyGen}(1^\lambda)$

$\mathsf{K} \leftarrow \mathsf{AKeyGen}(1^\lambda)$

$C^h(\mathsf{K}) \leftarrow \mathsf{HEnc}(\mathsf{pk}, \mathsf{K}) \xrightarrow{\mathsf{pk}, C^h(\mathsf{K})}$ $\mathsf{pk}, C^h(\mathsf{K})$

$C^a(M_i) \leftarrow \mathsf{AEnc}(\mathsf{K}, M_i) \xrightarrow{C^a(M_i)}$ $C^a(M_i)$

$$C^h(M_i) = \mathsf{CTC}\big(\mathsf{evk}, \mathsf{ADec}, C^h(\mathsf{K}), C^a(M_i)\big)$$

$f \xrightarrow{f}$ $C^h(f(M_0, \cdots, M_{\ell-1})) = \mathsf{HEval}(f, \mathsf{evk}, \{C^h(M_i)\})$

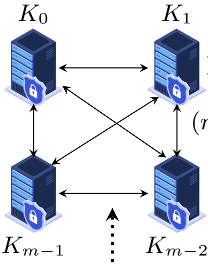
$f(M_0, \cdots, M_{\ell-1}) \xleftarrow{C^h(f(M_0, \cdots, M_{\ell-1}))}$

[1] This protocol is from [MJSC16]

Sharing IoT data among servers: data safe, single point of failure

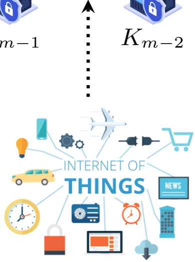
Settings:

$(K_0, \cdots, K_{n-1}) \leftarrow \mathsf{Sh}(K)$



Dist. sym-decryption:

$(m_0, \cdots, m_{n-1}) \leftarrow \mathcal{C}^{\mathsf{dd}}(\oplus K_i, c)$



Naive solution:

$(M_0, \cdots, M_{n-1}) \leftarrow \mathsf{Sh}(M)$

$C_i \leftarrow \mathsf{AEnc}(K, M_i)$

n encryption

Dist. decryption:

$C = \mathsf{AEnc}(K, M)$

1 encryption

Saving $n-1$ encryption

Fully homomorphic encryption (FHE) scheme [Gen09]

$$\mathsf{FHE} = (\mathsf{HKeyGen}, \mathsf{HEnc}, \mathsf{HDec}, \mathsf{HEval})$$

Key generation: $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{HKeyGen}(1^\lambda)$

Encryption: $c \leftarrow \mathsf{HEnc}(\mathsf{pk}, m)$

Decryption: $m \leftarrow \mathsf{HEnc}(\mathsf{sk}, c)$

Homomorphic evaluation:

- Add: $\mathsf{Add}(\mathsf{HEnc}(\mathsf{pk}, m_1), \mathsf{HEnc}(\mathsf{pk}, m_2)) = \mathsf{HEnc}(\mathsf{pk}, m_1 + m_2)$
- Mul: $\mathsf{Mul}(\mathsf{HEnc}(\mathsf{pk}, m_1), \mathsf{HEnc}(\mathsf{pk}, m_2)) = \mathsf{HEnc}(\mathsf{pk}, m_1 m_2)$
- Any function f

$$\mathsf{HEnc}(f(m_0, \cdots, m_{\ell-1})) \leftarrow \mathsf{HEval}(\mathsf{evk}, f, \{c_i\}_{i=0}^{\ell-1})$$

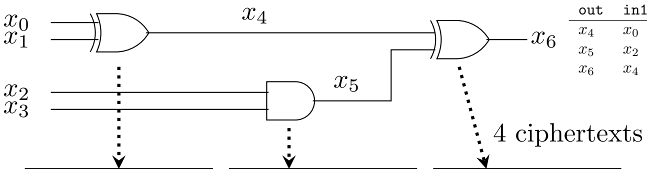
Homomorphic Encryption Standard[2]

- HomomorphicEncryption.org is an open consortium to standardize homomorphic encryption
  - industry, government, academia
- Goals
  - Unified and simplified API
  - Clear and understandable security properties
- Some open-source homomorphic encryption schemes [1]
  - HELib, Microsoft SEAL, PALISADE,
  - TFHE/FHEW, HeaAn, $\Lambda \circ \lambda$, NFLlib, cuHE, Lattigo

[2] https://homomorphicencryption.org

Garbled Circuit [Yao86]

Garbler: Garbling a circuit



$$\boxed{\begin{array}{c} E_{k_0^0}(E_{k_1^0}(k_4^0)) \\ E_{k_0^0}(E_{k_1^1}(k_4^1)) \\ E_{k_0^1}(E_{k_1^0}(k_4^1)) \\ E_{k_0^1}(E_{k_1^1}(k_4^0)) \end{array}} \quad \boxed{\begin{array}{c} E_{k_2^0}(E_{k_3^0}(k_5^0)) \\ E_{k_2^0}(E_{k_3^1}(k_5^0)) \\ E_{k_2^1}(E_{k_3^0}(k_5^0)) \\ E_{k_2^1}(E_{k_3^1}(k_5^1)) \end{array}} \quad \boxed{\begin{array}{c} E_{k_4^0}(E_{k_5^0}(k_6^0)) \\ E_{k_4^0}(E_{k_5^1}(k_6^1)) \\ E_{k_4^1}(E_{k_5^0}(k_6^1)) \\ E_{k_4^1}(E_{k_5^1}(k_6^0)) \end{array}}$$

Straight-line program

out	in1	in2	gate
x_4	x_0	x_1	XOR
x_5	x_2	x_3	AND
x_6	x_4	x_5	XOR

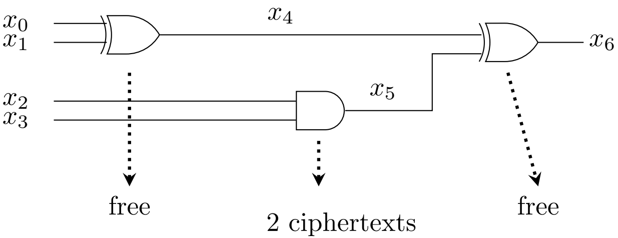
Evaluator: Evaluating a garbled circuit

- get correct input layer keys (k_i^b) using oblivious transfer (OT)
- correctly decrypt each garbled gate

Garbled Circuit (Cont.)

GC Optimizations: Permute-and-Point [BMR90], Row Reduction [NPS99], Free XOR [KS08], ..., Half Gates [ZRE15]

Using Half Gates:



 # AND in a circuit matters!

State-of-the-art: Sym-key & PEC

Stream ciphers for FHE

- FILP [MJSC16], Kreyvium [CCF+18], and Rasta [DEG+18]

Block ciphers for MPC & FHE

- LowMC [ARS+15], MiMC [AGR+16], GMiMC [AGP+19], and MARVELlous [AD18, AABS+20]

Hash functions for MPC & ZK Proofs

- GMiMC [AGP+19], and MARVELlous [AD18, AABS+20], Poseidon [GKR+19]

Our Results

Binary Circuits for LWC Candidates

Why binary circuits for lightweight ciphers?

- SMPC protocols, FHE schemes, ZK proofs

AE core of an AE Scheme: Nonlinear primitive

- Nonlinear primitive + Linear logic for mode
- Underlying permutation is the AE core of a sponge-based AE

Different AE cores in round 2 candidates

- 24 different AE cores

Binary Circuits for LWC Candidates

Candidates	**Core-primitive**	**Type**
ACE	ACE	Permutation
COMET, ESTATE, mixFeed, SAEAES	AES	Block cipher, Tweakable Block cipher
ASCON, ISAP	ASCON	Permutation
COMET	CHAM	Block cipher
DryGASCON	DryGASCON	Permutation
ESTATE, GIFT-COFB, HyENA, LOTUS-AEAD, LOCUS-AEAD, SUNDAE-GIFT	GIFT-64/128	Block cipher, Tweakable block cipher
Gimli	Gimli	Permutation
Grain-128AEAD	Grain	Stream cipher
ISAP, Elephant	Keccak	Permutation
KNOT	Knot	Permutation
ORANGE, PHOTON-Beetle	Photon	Permutation
Oribatida	SimP-n-θ	Block cipher
Pyjamask	Pyjamask	Block cipher
Saturnin	Saturnin	Block cipher
SPIX, SpoC	sLiSCP-light-192/256	Permutation
Sparkle	Sparkle	Permutation
COMET	Speck	Block cipher
Elephant	Spongent	Permutation
Spook	Clyde-128 and Shadow-512	Tweakable block cipher, Permutation
ForkAE, Romulus, SKINNY-AEAD,	Skinny	Block cipher
Subterranean	Subterranean	Permutation
TinyJambu	TinyJambu	Stream cipher
WAGE	WAGE	Permutation
Xoodyak	Xoodyak	Permutation

Binary Circuit Stat of AE Cores

Generating Boolean circuits of AE cores

-Using the CBMC-GC compiler [FHK+14]

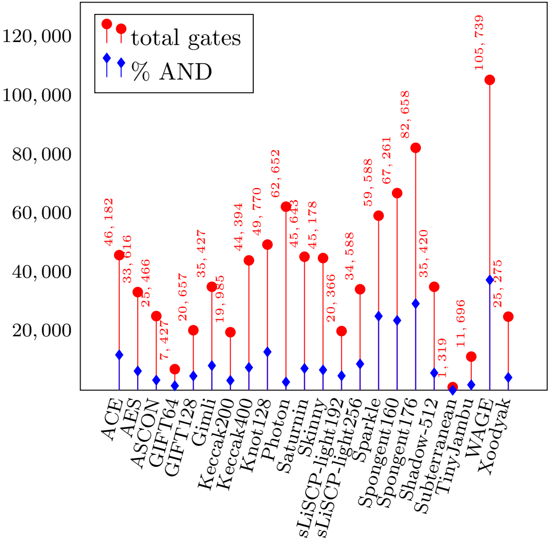
Gates:
XOR, AND, NOT

AND %

Photon: %7.02

Sparkle: %42.69

WAGE: %35.70



Evaluating AE Cores

Task 1: Secure evaluation of AE cores in MPC

- Securely evaluating the AE cores in the two-party computation
- Garbled circuit

Task 2: Homomorphic evaluation of AE cores

- Homomorphically evaluating the AE cores using a FHE scheme
- FHE works over binary circuits (e.g., TFHE)

Our Settings for AE Core Evaluations

Secure evaluation of core AE circuits

State: S Key: K Nonce: N

Garbler

$$S = S_1 \oplus S_2$$
$$K = K_1 \oplus K_2$$
$$N = N_1 \oplus N_2$$

Evaluator

$\mathcal{C}$

Inputs: K_1, N_1 K_2, N_2

$$S_1 \oplus S_2 \leftarrow \mathcal{C}(K_1 \oplus K_2, N_1 \oplus N_2)$$

Garbled circuit

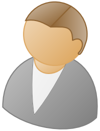
Outputs: S_1 S_2

Boolean Message Sharing in Sponge

Functionality: Boolean sharing of message blocks

Circuit $\mathcal{C}^{\mathsf{bms}}(S, R_i, C_i)$:

1. $(S_r, S_c) = S \leftarrow \mathcal{C}(S)$
2. $RK_i \leftarrow S_r \oplus R_i$
3. $M_i^1 \leftarrow R_i$, $M_i^2 \leftarrow C_i \oplus RK_i$
4. $M_i \leftarrow M_i^1 \oplus M_i^2 = R_i \oplus C_i \oplus S_r \oplus R_i$



Common: S, C_i

Input: R_i

$(M_i^1, M_i^2) \leftarrow \mathcal{C}^{\mathsf{bms}}(S, R_i, C_i)$

$M_i^2 \leftarrow C_i \oplus RK_i$

Output: M_i^1

M_i^2

Boolean Message Sharing in Sponge

Init State: $S = K \| N = S^1 \oplus S^2$, Ctxt: $C = (C_0, \cdots C_{\ell-1})$

Garbler **Evaluator**

$(M_0^1, \cdots M_{\ell-1}^1)$

$(M_0^2, \cdots M_{\ell-1}^2)$

Boolean Message Sharing in Sponge

Init State: $S = K \| N = S^1 \oplus S^2$, Ctxt: $C = (C_0, \cdots C_{\ell-1})$

Garbler **Evaluator**

$(M_0^1, \cdots M_{\ell-1}^1)$

$(M_0^2, \cdots M_{\ell-1}^2)$

$\mathcal{C}^{\mathsf{bms}}$

Input: S^1, R_0 S^2

$(M_0^1, M_0^2) \leftarrow \mathcal{C}^{\mathsf{bms}}(S^1 \oplus S^2, R_0)$

Output: $M_0^1 = R_0$ **Garbled circuit + OT** M_0^2

$\vdots$ $\vdots$

Input: $R_{\ell-1}$ $(M_1^1, M_1^2) \leftarrow \mathcal{C}^{\mathsf{bms}}(S^1 \oplus S^2, R_{\ell-1})$

Output: $M_{\ell-1}^1 = R_{\ell-1}$ **Garbled circuit + OT** $M_{\ell-1}^2$

Experimental Evaluation

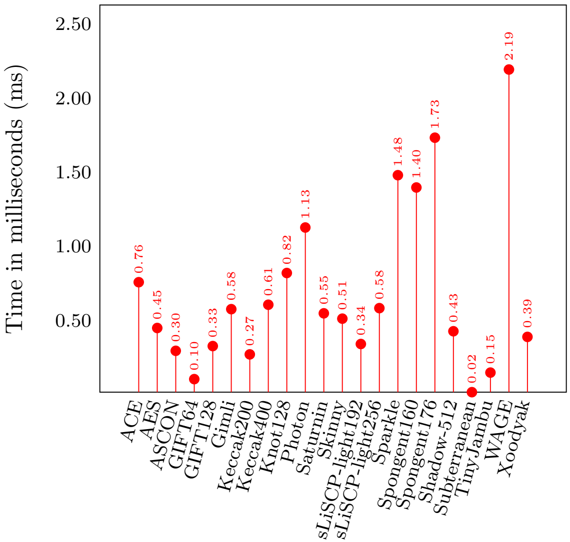
Settings:

- Considering 2-server settings
- Evaluating AE core circuits
- Use garbled circuit for 2PC computation

Implementation:

- Develop a generic implementation in C++ on top of the EMP-toolkit libraries [WMK16]
- Executing codes on a desktop with 3.00GHz Intel Coffee lake CPU, and 32 GB RAM running on Ubuntu 18.04

Timing for 2PC AE Cores Evaluation

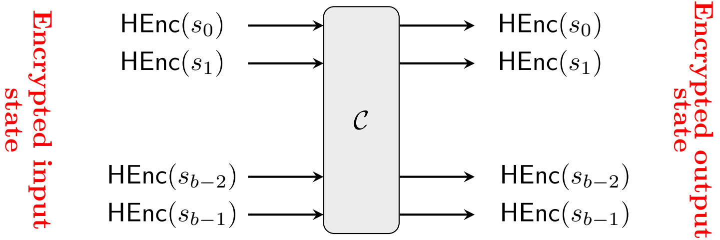


Homomorphic Evaluation of AE Cores

Initialize State: $(s_0, \cdots, s_{b-1}) = S = K \| N$

Encrypted key: $\mathsf{HEnc}(K) = (\mathsf{HEnc}(k_0), \cdots, \mathsf{HEnc}(k_{n-1}))$

$\mathcal{C}$ = AE core circuit

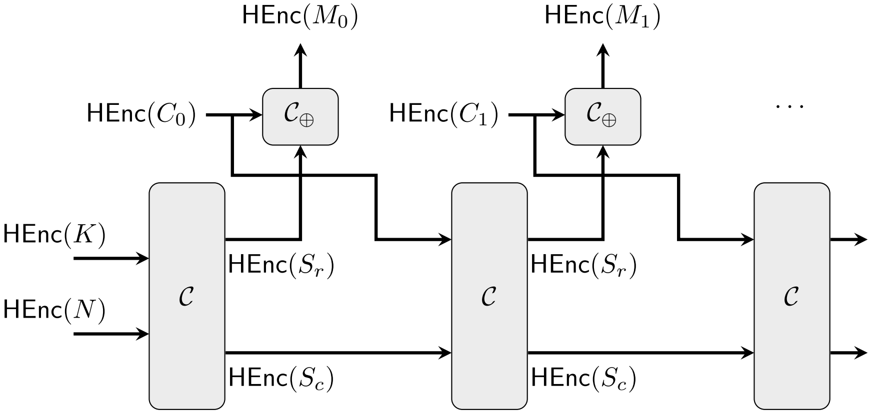


Homomorphically evaluating

Homomorphically Evaluating Sponge

State $S = (S_r, S_c)$ **Ctxt** $C = (C_0, \cdots), C_i = \mathsf{AEnc}(K, M_i)$

Compute: $\mathsf{HEnc}(M_i)$ from C_i where $C_i = \mathsf{AEnc}(K, M_i)$



Experimental Evaluation

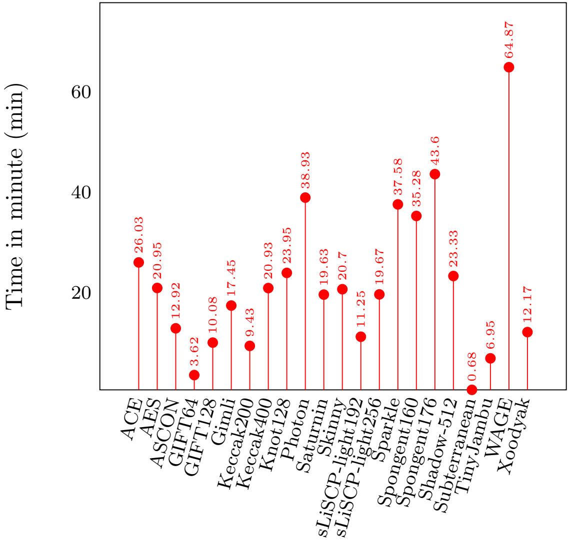
Settings:

- Homomorphically evaluating AE core circuits
  - Encrypt input state
  - Obtain encrypted output state
- Use the TFHE scheme [CGGI16]
  - Efficient for Boolean operations such as XOR, AND and OR
  - Gate-level bootstrapping $\Rightarrow$ circuit depth is not an issue

Implementation:

- Develop a generic implementation in C++ on top of the TFHE library
- Executing codes on a desktop with 3.00GHz Intel Coffee lake CPU, and 32 GB RAM running on Ubuntu 18.04

Time for AE Cores Eval using TFHE



Estimating Cost for Overall Mode

Estimating time for individual modes

- Using AE core evaluation cost + linear operation costs from micro-benchmark results

	Number of ciphertexts		
Operation	64	128	256
TFHE.Enc	0.002457 s	0.00490 s	0.010055 s
TFHE.Dec	0.000065 s	0.00013 s	0.000266 s
TFHE.XOR	2.502675 s	4.959481 s	10.04321 s
TFHE.AND	2.476224 s	4.964595 s	9.981702 s
TFHE.NOT	0.000058 s	0.000118 s	0.000231 s

Summary

- Secure evaluation of lightweight ciphers for privacy-enhancing cryptographic applications

- Optimized Boolean circuits of some core primitives of NIST LWC round 2 candidates

- Privacy-preserving evaluation of AE core circuits
  - 2PC evaluation: garbled circuit
  - FHE evaluation: TFHE

This work is in progress ...

Thanks for your attention!

Questions?