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, CANADA

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

Abstract. Motivated by a number of applications of lightweight ciphers in privacy-enhancing cryptography (PEC) techniques such as secure multiparty computation (SMPC) and fully homomorphic encryption (FHE), we investigate the Boolean circuit complexity of the core primitives of NIST lightweight cryptography (LWC) round 2 candidates. In PEC, the functionalities (e.g., ciphers) are often required to express as Boolean or arithmetic circuits before applying PEC techniques, and the size of the circuit is one of the efficiency factors. As use cases, we consider secure evaluation of the core AE circuits in the two-party computation (2PC) setting using Yao garbled circuit, and homomorphic evaluation of the core AE circuits in the cloud-outsourcing setting using the TFHE scheme. The performance results for both cases are presented.

1 Introduction

Pervasive and ubiquitous computing has been integrating the physical world into the digital world where billions of physical devices such as smart devices, sensors and actuators are deployed at different applications for operational, monitoring and data collection. The collected data are processed at backend servers/cloud for operation, automation, and optimizing costs. Despite the use of lightweight cryptographic algorithms such as authenticated encryption (AE) for protecting communication in resource-constrained applications, it may be used for secure storage of data from constrained applications. Such stored data will not solely be used for storage or backup (securing while at rest) purposes, but also be used for performing analytics computations for extracting useful information for operational, automation, and optimizing cost purposes. To enable computation on encrypted data by a lightweight cipher, a friendly (computationally-efficient) interface of the lightweight cipher with privacy-enhancing cryptography (PEC) techniques such as secure multiparty computation (SMPC) and fully homomorphic encryption (FHE) is required.

The idea of friendliness of a symmetric-key algorithm with SMPC protocols or FHE schemes or bridging the gap between a symmetric-key algorithm and a public-key algorithm using key/data encapsulation mechanisms is not new [GRR+16, MJSC16, AGR+16, DEG+18, CS03,

*A preliminary result is submitted to the NIST Lightweight Cryptography Workshop 2020 from an ongoing work.

CCF+18]. In the literature of SMPC, FHE and zero-knowledge (ZK) proof, AES has not only been widely used as a benchmarking cipher, but also used to develop privacy-preserving applications using secure computation techniques [DK10, DKL+12, KOR+17, LR15, PSSW09, NNOB12, GHS12, LTW13, KMR12]. Block ciphers namely Triple-DES and SIMON are also used as benchmarking ciphers for SMPC and FHE applications [LN14, KOR+17]. Symmetric-key primitives are also used in cryptocurrency and blockchain applications along with zero-knowledge proof techniques. For example, Zcash [SCG+14], the latest version (2019) adopted CHACHA20_POLY1305 [Hou17] for authenticated encryption, and the BLAKE2 hash function (the top 2 finalist in the NIST SHA3 competition for hash). Hawk [KMS+16] uses lightweight block cipher Speck [BSS+15] for encryption in the CBC mode, and SHA-256 for pseudorandom functions (PRFs) and commitments, and achieves only 80-bit security.

The ongoing NIST lightweight cryptography (LWC) standardization competition has aimed at standardizing lightweight authenticated cipher(s) and hash function(s) [BCC+19]. Although lightweight authenticated ciphers have aimed at providing security in constrained environments, their real-world deployment would not only be limited to resource-constrained environments, rather be used at heterogeneous computing environments, depending upon their efficiencies. Therefore, the friendliness of lightweight ciphers with PEC techniques should be evaluated. In this work, we consider secure evaluation of lightweight ciphers in privacy-enhancing cryptographic applications.

1.1 Applications of Lightweight AE in Privacy-enhancing Cryptography

In this section, we mention some applications where privacy-enhancing aspects of lightweight authenticated ciphers are required to consider. We use the Internet of Things (IoT) as an application to explain the usages.

1.1.1 Cloud-IoT Outsourcing Data and Computation for Analytics

Smart IoT devices are deployed for operational and automation at various applications. Consider a scenario where a smarthome thermostat sends temperature readings of a house in every an hour basis to a cloud, who processes the temperature readings and instruct the thermostat to automate the heating in the house. Note that cleartext temperature readings leaks information about individuals' presence or absence in the house. Cloud can do the profiling of houses from temperature readings, which invades privacy. This can be prevented by a conjunction of lightweight AE algorithms with FHE algorithms. Assume the thermostat has implemented a lightweight AE scheme. For each day, it uses a nonce that can be a time stamp of the day and encrypt each hour's temperature reading using a block of AE encryption where a temperature reading can be represented using a 32, 64 or 128 bit number. Thus, for each day 24 readings are sent to the cloud, and by the end of the day, it converts to AE ciphertexts to HE ciphertexts, and then performs statistical/analytical computations on encrypted readings. One of the key consideration is an efficient integration of AE and FHE algorithms (see Section 2.3 for details).

1.1.2 Sharing of Data from Constrained Applications to the MPC System

In an MPC system, data from multiple sources are secret-shared among a set of MPC servers, denoted by S_i, who jointly conduct the computation over secret-shared data to obtain the result. Assume there are ℓ MPC servers in an MPC system. We consider a new constrained-environment

application scenario where data is encrypted by an IoT device and the ciphertext is denoted by $c = \mathsf{AEnc}(k, m)$ and the encryption key k is secret-shared among the MPC servers, i.e., $k = \bigoplus_{i=1}^{\ell} k_i$. To optimize the cost, the IoT device encrypts the data only once and sends it to the MPC system. To create secret-shares of the data m, the MPC servers jointly need to evaluate the decryption algorithm, and this functionality is described as follows:

Inputs of S_i: Server S_i holds a share of the key k_i and randomly generates r_i for $1 \leq i \leq \ell - 1$. S_ℓ gives no random input.

Jointly generate the share of data: $(m_1, m_2, \cdots, m_\ell) \leftarrow f((k_1, r_1), \cdots, (k_{\ell-1}, r_{\ell-1}), (k_n), \mathsf{ADec}, c)$ such that $\sum_{i=1}^{\ell} m_i = m$, which consists of the following steps: a) Compute $m = \mathsf{ADec}(\bigoplus k_i, c)$, b) Compute $m_\ell = m \oplus (\bigoplus r_i)$. By the end of the computation, for the servers' $S_i, 1 \leq i \leq \ell - 1$, the share is $m_i = r_i$,, and the server S_ℓ's share is m_ℓ.

This requires the distributed decryption of an AE scheme which saves $(\ell - 1)$ encryption operations for the IoT device, over encrypting and sending individual ciphertexts of $m_i, 1 \leq i \leq \ell$ by the device. After the secret-sharing of data m, the servers' can perform computations on data using the prescribed MPC protocol. As mentioned in [ARS+15], the MPC server can jointly encrypt secret information to send it to the IoT device. This motivates us to consider the circuit complexity of the lightweight AE schemes. The ideas about jointly encrypting by the MPC servers for AES or other PRF-based AE have appeared in [RSS17, ARS+15].

1.1.3 Secure Password Storage, IoT Database Operation and More

A number of applications for secure evaluation of AES such as one-time passwords [ARS+15], secure database join operations [LTW13], and private set intersection [PSSW09] have been proposed. For more collected applications of AES, the reader is referred to [ARS+15]. When a lightweight cipher is standardized, it is likely that the lightweight cipher would also be used for various applications. Many of these privacy-preserving applications may be directly adapted to resource-constrained applications such as IoT databases [1][2].

1.2 Related Work

Lightweight cryptography and NIST LWC standardization competition. The advent of lightweight cryptography is due to providing security in resource-constrained environments such as RFID and sensors where traditional ciphers for computer/internet may be heavy [BKL+07]. Lightweight cryptography exists for more than a decade. There have been numerous symmetric-key ciphers such as block ciphers, stream ciphers, authenticated encryption and hash functions developed in the past years targeting to hardware efficiency. Some notable examples are PRESENT, CLEFIA, and LEA in the ISO/IEC standards [Sta19], Grain and Trivium from the eStream project [ECR], and ASCON and ACORN from the CAESAR competition [CAE]. In response to the call for proposal of the NIST LWC standardization competition, there were 56 proposals for authenticated encryption (many with both AE and hash) accepted as round 1 candidates, and 32 candidates were moved to round 2 [BCC+19]. The design of lightweight AE schemes can be classified as four main types namely permutation, block cipher, tweakable block cipher, stream cipher and some other designs [STMÇ+19].

[1] `https://www.ibm.com/downloads/cas/G6BONV4B`

[2] `https://crate.io`

Symmetric-key ciphers for privacy-enhancing cryptography. There is a growing interest in the development of symmetric-key ciphers dedicated to privacy-enhancing cryptographic applications such as secure multiparty computation, fully homomorphic encryption and zero-knowledge proofs. Some examples of stream ciphers designed for FHE applications are FILP [MJSC16], Kreyvium [CCF+18], and Rasta [DEG+18], the block cipher examples for MPC, FHE and ZK proof applications include LowMC [ARS+15], MiMC [AGR+16], GMiMC [AGP+19], and MARVELlous [AD18, AABS+19]. Examples of hash functions for MPC and ZK proofs applications include GMiMC [AGP+19], MARVELlous [AD18, AABS+19], and Poseidon [GKR+19]. [RSS17] presents the constructions of parallel nonce-based authenticated encryption based on the MiMC and Legendre symbol PRFs for MPC applications. To the best of our knowledge, the suitability of these primitives, especially block ciphers and stream ciphers have not been investigated for resource-constrained applications where the hardware efficiency is a major consideration.

For blockchain and cryptocurrency applications, Boolean circuits of LWC also has very important applications in privacy of blockchain systems, especially the zero-knowledge Succinct Non-interactive Argument of Knowledge (zkSNARK) for blockchain privacy [GGPR13]. Even for most advanced servers, it is harder to perform these computations in practice at an adequate security level. In fact, a few years ago, only a few blockchain and cryptocurrency systems have implemented privacy protection mechanisms by zkSNARKs (e.g., Zcash [SCG+14], Hawk [KMS+16] and Ligero [AHIV17]), but at security levels below 128 bits. These applications point out that there is a gap between the efficient interface of lightweight cryptography and privacy-enhancing cryptography as the applications of lightweight ciphers may go beyond traditional encryption and authentication due to increasing privacy concerns.

1.3 Our Contribution

This work focuses on engineering and real-world application aspects of lightweight ciphers when conjunct with privacy-enhancing/preserving cryptographic techniques such as SMPC and FHE. First, we generate the Boolean circuits of core primitives of the NIST LWC AE schemes, and present some empirical circuit complexity statistics. The reason for studying the circuit complexity is that in PEC, the functionalities (e.g., ciphers) are often required to express as Boolean or arithmetic circuits before applying PEC techniques, and the running time (efficiency) of the privacy-preserving scheme relies on the size of the circuit. Due to the heterogeneity of the AE modes in the NIST LWC round 2 candidates, we consider the core underlying primitives of the AE schemes, which are the nonlinear components that are the bottleneck in PEC. Next, we consider the secure evaluation of core primitives in the standard 2PC computation setting using Yao garbled circuit (GC) [Yao86] and the homomorphic evaluation of AE core primitives in the computation-outsourcing setting. We developed two implementations in C++ on top of the EMP-toolkit [WMK16] for 2PC evaluation where the GC scheme is instantiated the half-gates garbling scheme [ZRE15] and the TFHE scheme [CGGI16a, CGGI16b] for homomorphic evaluation. Finally, we present the experimental results on the performance of the core AE primitives for both cases.

2 Preliminaries

2.1 Authenticated Encryption

An authenticated encryption with associated data (AEAD) scheme is a tuple of algorithms $\mathsf{AEAD} = (\mathsf{AKeyGen}, \mathsf{AEnc}, \mathsf{ADec})$. The key generation $\mathsf{AKeyGen}$ outputs a key, i.e., $\mathsf{K} \leftarrow \mathsf{AKeyGen}(1^\lambda)$ for security parameter λ, AEnc accepts a key K, a nonce N, an associated data AD and a message M and produces a ciphertext and a tag, i.e., $(C, T) \leftarrow \mathsf{AEnc}(K, N, AD, M)$. Similarly, ADec accepts a key, a nonce, an associated data, a ciphertext and a tag and produces a message if the tag verification is successful, i.e., $\{M, \perp\} \leftarrow \mathsf{ADec}(K, N, AD, C, T)$. The encryption algorithm AEnc has four phases, namely an initialization phase, an associated data processing phase, an encryption phase and a tag generation phase, and similarly for the decryption phase where the encryption phase is replaced by a decryption phase.

2.2 Two-party Computation Protocol and Garbled Circuit

Secure multiparty computation (SMPC) protocol allows a set of mutually distrusting parties to jointly compute a function on their private inputs without revealing anything information about the private inputs, except what directly leaks from the output. Secure 2-party computation (2PC) protocol is a special case of an SMPC protocol which is between two parties. Yao garbled circuit (GC) [Yao86] is a popular 2PC protocol where the functionality to be evaluated on the private inputs is represented as a Boolean circuit. For the details, the reader is referred to [Yao86, HL10, BHR12].

2.3 Fully Homomorphic Encryption and Outsourcing Protocol

Fully homomorphic encryption. Fully homomorphic encryption (FHE) is a public-key encryption scheme that enables to perform (in principle) an unlimited number computation over encrypted data for any arbitrary function represented using a circuit. An FHE scheme consists of a tuple of four probabilistic polynomial-time algorithms $\mathsf{FHE} = (\mathsf{HKeyGen}, \mathsf{HEnc}, \mathsf{HDec}, \mathsf{HEval})$ [Gen09]. The key generation algorithm $\mathsf{HKeyGen}$ generates secret, public, and evaluation keys, i.e., $(\mathsf{pk}, \mathsf{sk}, \mathsf{evk}) \leftarrow \mathsf{HKeyGen}(1^\lambda)$ for a security parameter λ, HEnc encrypts a plaintext message (m) using the public key, i.e., $c \leftarrow \mathsf{HEnc}(\mathsf{pk}, m)$, HDec decrypts a ciphertext (c) using the private key, i.e., $m \leftarrow \mathsf{HEnc}(\mathsf{sk}, c)$, and HEval evaluates a function f (typically represented using a circuit) on a set of ciphertexts $(\{c_i\}_{i=0}^{\ell-1})$ using the evaluation key (evk), and produces a single ciphertext $\mathsf{HEnc}(f(m_0, \cdots, m_{\ell-1})) \leftarrow \mathsf{HEval}(\mathsf{evk}, f, \{c_i\}_{i=0}^{\ell-1})$ which is the encrypted output of f on plaintext messages $(\{m_i\}_{i=0}^{\ell-1})$.

Homomorphic encryption security standard. Recent years have evidenced a promising advancement of FHE schemes, open-source implementations and applications and growing demands from industry. There is a homomorphic encryption standard initiated to standardize FHE scheme(s) to have a unified and simplified API, and clear and understandable security properties for use by non-experts as well as experts. For the details about the standard and the list of FHE candidates, the reader is referred to [ACC+18, ea18b, ea18a].

Hybrid-encryption based outsourcing protocol. Figure 1 shows a computation and data outsourcing protocol combining a symmetric-key encryption scheme and a fully homomorphic encryption scheme. It follows the paradigm of key and data encapsulation mechanisms

(KEM/DEM) for hybrid encryption where the data is encrypted using a symmetric encryption and the key of the symmetric-key encryption is encrypted using a public key scheme (FHE), i.e., $KEM \| DEM = \mathsf{HEnc}(\mathsf{pk}, \mathsf{K}) \| \mathsf{AEnc}(\mathsf{K}, Data)$ [CS03]. Note that the client generates the keys for both homomorphic encryption and authenticated encryption schemes. The ciphertext conversion step from a symmetric-key to a HE ciphertext, denoted by CTC, needs the homomorphic evaluation of the symmetric-key decryption algorithm. In this work, our main focus is on the ciphertext conversion algorithm when the symmetric-key encryption algorithm is a lightweight AE scheme.

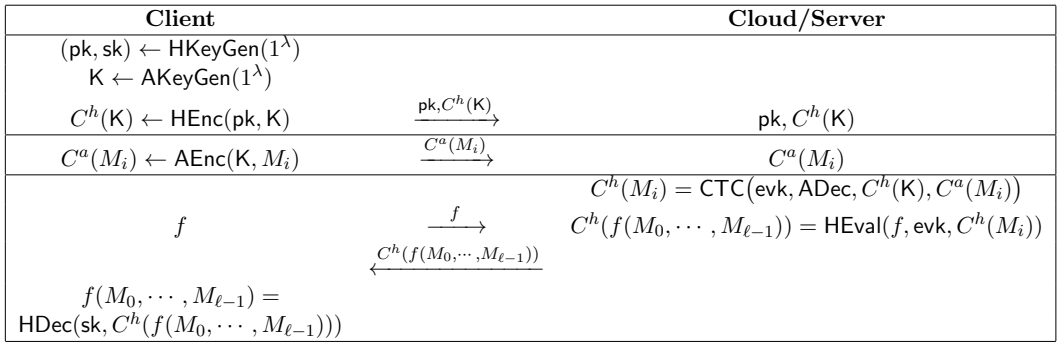

Figure 1: An FHE-based client-server computation and data outsourcing protocol using symmetric-key encryption [MJSC16].

3 Circuit Complexity of NIST LWC Round 2 Candidates

In this section, we generate and report the Boolean circuits of the core primitives of the NIST LWC round 2 candidates. The reason for this is that, in many privacy-enhancing applications, the functionalities are required to represent as a Boolean circuit before applying the privacy-enhancing techniques.

Classification of round 2 AE candidates into core primitives. We call a component of an AE scheme a *core primitive* if it is the nonlinear component of the AE scheme. For instance, for a permutation-based AE scheme, the core primitive is the permutation as it provides the nonlinearity and the mode part involves linear operations. The underlying core primitives of the AE schemes can be classified according to the main primitives as shown in Table 1. Note that, for example, the block cipher GIFT-128 has been used as a core primitive in multiple submissions, we consider only the GIFT-128 block cipher as it is the core nonlinear component of the schemes. As there are various types of modes in the NIST LWC candidates, we consider only the core primitives in the AE schemes.

Generating Boolean circuits of core primitives. We generate the circuits for the underlying permutations, block ciphers or state update functions of the AE or hash modes. We use the CBMC-GC compiler [FHK+14] to generate the circuits where each LWC cipher's circuit is represented using XOR, AND and NOT gates in the Bristol fashion [bri]. Table 2 summarizes

Table 1: Classification of the MIST round 2 candidates based on the underlying core primitives

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

the list of circuits with the numbers of XOR, AND and NOT gates[3]. The multiplicative depth and the total depth of the LWC circuits are also reported. The description of the circuits can be found in [Man20]. We do not claim that the circuit sizes are minimal.

Multiplicative depth. The multiplicative depth of a Boolean circuit is the maximum number of sequential multiplications (AND operations) in the circuit. As the construction of the core primitive is iterative and based on a round function consisting of linear and nonlinear functions, the multiplicative depth of the primitive can be easily seen from the number of rounds and the multiplicative depth of the round function (Proposition 1). For instance, for ACE, the Feistel

[3]The table does not contain the details of all round 2 candidates. The work of the remaining candidates is in progress.

round function in the Simeck-box is a quadratic function of an AND depth 1 and the total of rounds is 128, the multiplicative depth is 128, which can be seen in the AND depth column of Table 2. Similarly, for ASCON, the round is quadratic, thus the multiplicative depth for the 12-round permutation is 12. The multiplicative depth of the Boolean circuits of the core primitives is an important consideration for the FHE applications where for some FHE scheme, the key setup parameters are chosen based on this information. Moreover, the noise growth due to the multiplication operation is larger than the noise growth due to that of the addition operation.

Proposition 1. *Let d be the multiplicative depth of a Boolean circuit of a round function composed of a substitution-permutation network (SPN) or Feistel network based cipher. The multiplicative depth of a r-round cipher is rd where r is the number of rounds.*

Proof. The proof is straightforward. □

Table 2: Summary of the circuit complexity of some NIST LWC core primitives in round 2.

Cipher	**State Size**	**Total Gates**	**Individual Gates**			**AND Depth**	**Total Depth**	**% AND**
			AND	XOR	NOT			
AES [TS]	128	33616	6800	25124	1692	–	–	20.23
ACE	320	46182	12288	27648	6246	128	475	26.61
Ascon	320	25466	3712	15932	5822	12	93	14.58
Ascon(r6)	320	12408	1792	7868	2748	6	47	14.58
Ascon(r8)	320	16760	2432	10556	3772	8	62	14.58
Gimli	320	35427	8640	17760	9027	24	75	24.39
GIFT-128	128	20657	5120	10240	5297	160	449	24.79
TweGIFT-64	64	20298	10315	8166	1817	56	196	50.82
TweGIFT-64-inv	64	19846	10315	7718	1813	56	196	51.98
Keccak-200	200	19985	3600	10800	5585	18	174	18.01
Keccak-400	400	44394	8000	24000	12394	20	192	18.02
Knot-256	256	49770	13312	23296	13162	104	260	26.75
Knot-384	384	109140	29184	51041	28980	152	380	26.74
Knot-512	512	191665	51200	89600	50865	200	500	26.71
Photon	256	62652	17940	41640	3072	24	179	28.63
Saturnin	256	45643	7680	22465	15627	120	331	16.83
Skinny-enc-128-384††	128	207506	65344	129215	12947	392	1462	31.49
Skinny-dec-128-384††	128	139699	41936	80573	17190	286	1095	30.02
sLiSCP-light-192	192	20366	5184	12096	3086	108	437	25.45
sLiSCP-light-256	256	34588	9216	20736	4636	144	542	26.65
sLiSCP-light-256 (r9)	256	17324	4608	10368	2348	72	271	26.65
Sparkle-256	256	59588	25440	31360	2788	200	554	42.69
Sparkle-384	384	98422	41976	51920	4526	220	613	42.65
Sparkle-512	512	143524	61056	75648	6820	240	703	42.54
Spongent-160	160	72890	20211	41321	11358	160	534	27.73
Spongent-176	176	93429	26702	52396	14331	180	646	28.58
Spook: Shadow-512	512	35420	6144	29184	92	19	94	17.35
Clyde-128-Enc	128	13655	1536	12096	23	24	132	11.25
Clyde-128-Dec	128	13655	1536	12096	23	37	161	11.25
Subterranean	257	1319	265	772	290	2	7	20.09
TinyJambu-init†	128	11696	2118	8422	1156	50	276	18.11
TinyJambu (P_{1024})†	128	5638	1024	4096	518	24	134	18.16
WAGE	259	105739	37745	62121	5873	333	2220	35.70
Xoodoo	384	25275	4608	13824	6843	12	93	18.23

† = 112-bit security, †† = reduced # of gates, not low AND-depth circuits.

4 Secure 2PC Evaluation of Core AE Circuits

Motivated by applications of the lightweight AE schemes in MPC in Section 1.1, we evaluate the performance of lightweight core primitives using Yao garbled circuit, and present the experimental results in the semi-honest adversarial settings.

Securely evaluating core AE circuits. We consider the problem of securely evaluating the core primitive of an AE scheme, which is the main computationally-expensive component. The total evaluation time of the mode of an AE scheme can be estimated from the core primitive evaluation time by including the cost of oblivious transfer operations to compute each block of the messages. For now, we restrict ourselves to the evaluation of the core primitives. For a core AE circuit $\mathcal{C}$, we evaluate $\mathcal{C}(K_1 \oplus K_2, M_1 \oplus M_2)$ using the garbled circuit where one party holds the input (K_1, M_1), another party holds (K_1, M_1), and $K = K_1 \oplus K_2$ and $M = M_1 \oplus M_2$, which are Boolean sharings of K and M, respectively. For instance, for a block cipher circuit $\mathcal{C}$, $\mathsf{AEnc}(K_1 \oplus K_2, M_1 \oplus M_2) = \mathcal{C}(K_1 \oplus K_2, M_1 \oplus M_2)$.

Implementation details. We develop a generic implementation in `C++` on top of the EMP-toolkit libraries [WMK16] that implemented the oblivious transfer (OT) protocol and the half-gates garbling scheme [ZRE15]. We consider the garbled circuit scheme that is secure against semi-honest adversaries. In our implementation, we feed the circuits generated in Section 3 and obtain the computational time for both garbling and evaluation of the core circuits. Note that no network communication was involved during the execution of the protocol as both the garbler and evaluator were running on the same machine.

Performance. The experiments were conducted on a desktop with 3.00GHz Intel Core i7-9700 CPU and 32 GB RAM running on Ubuntu 18.04. Table 3 presents the wall-clock time to evaluate one execution of the core primitive circuit. Note that the garbler and the evaluator's time are similar. Thus we present only the evaluator time in the table.

5 Homomorphic Evaluation of Core AE Circuits

As shown in Section 2.3, converting an AE ciphertext to an FHE ciphertext involves the homomorphic evaluation of the core AE circuits. In this section, we perform the homomorphic evaluation of the core AE circuits of the NIST LWC AE schemes and present experimental results. Before we present the experimental evaluation results, we explain the process of converting a ciphertext produced by an AE scheme to a ciphertext of an FHE scheme.

5.1 Conversion of AE Ciphertexts to FHE Ciphertexts

We use a sponge-based AEAD as an example to explain the process and focus only on the encryption and decryption process. Let S be the state of the permutation π after initialization and associated data processing phases. Assume the encryption of a message is performed, like a stream cipher encryption, as $C_i = M_i \oplus K_i$ where K_i is served as a keystream block that is obtained from the rate part of the state of the permutation after processing previous $(i-1)$ message blocks, i.e., $K_i \leftarrow \lfloor \pi^i(S) \rfloor_r$ where $\lfloor \cdot \rfloor_r$ denotes the contents from the rate part. Assume that a cloud receives ciphertexts $C = (C_0, \cdots, C_{\ell-1})$ and the encrypted key of the AE scheme,

Table 3: Timing is for the evaluator's running time. Timings are in millisecond (ms).

	ACE	AES	Ascon	Gimli	GIFT-128	TweGIFT-64	TweGIFT-64-inv
Total Time	0.758 ms	0.449 ms	0.295 ms	0.576 ms	0.327 ms	0.565 ms	0.558 ms
	Keccak-200	Keccak-400	Knot-256	Knot-384	Knot-512	Photon	Saturnin
Total Time	0.271 ms	0.606 ms	0.820 ms	1.867 ms	3.140 ms	1.127 ms	0.548 ms
	Skinny-enc	Skinny-dec	sLiSCP-light-192	sLiSCP-light-256	Sparkle-256	Sparkle-384	Sparkle-512
Total Time	3.764 ms	2.510 ms	0.341 ms	0.583 ms	1.480 ms	2.461 ms	3.611 ms
	Spongent-160	Spongent-176	Shadow-512	Clyde-128-Enc	Clyde-128-Dec	Subterranean	TinyJambu-init
Toral time	1.239 ms	1.597 ms	0.427 ms	0.121 ms	0.121 ms	0.016 ms	0.148 ms
	TinyJambu (P_{1024})	WAGE	Xoodoo	Ascon (r6)	Ascon (r8)	sLiSCP-light-256 (r9)	
Total Time	0.070 ms	2.193 ms	0.389 ms	0.141 ms	0.190 ms	0.287 ms	

Table 4: Homomorphic evaluation times of core AE circuits using TFHE with security 110 bits. Timings are given in second (s), and the fractional part is omitted.

	ACE	AES	Ascon	Gimli	GIFT-128	TweGIFT-64	TweGIFT-64-inv
Total Time	1562 s	1257 s	775 s	1047 s	605 s	732 s	716 s
	Keccak-200	Keccak-400	Knot-256	Knot-384	Knot-512	Photon	Saturnin
Total Time	566 s	1256 s	1437 s	3144 s	5533 s	2336 s	1178 s
	Skinny-enc	Skinny-dec	sLiSCP-light-192	sLiSCP-light-256	Sparkle-256	Sparkle-384	Sparkle-512
Total Time	7936 s	5113 s	675 s	1180 s	2255 s	3663 s	5346 s
	Spongent-160	Spongent-176	Shadow-512	Clyde-128-Enc	Clyde-128-Dec	Subterranean	TinyJambu-init
Toral time	2688 s	3425 s	1400 s	542 s	542 s	41 s	417 s
	TinyJambu (P_{1024})	WAGE	Xoodoo	Ascon (r6)	Ascon (r8)	sLiSCP-light-256 (r9)	
Total Time	203 s	3892 s	730 s	381 s	507 s	588 s	

i.e., $\mathsf{HEnc}(K\|N)$ where $C = \mathsf{AEnc}(K, N, M)$. The cloud performs the following steps to obtain $\mathsf{HEnc}(M)$ as follows:

Step 1: Compute an FHE ciphertext from an AE ciphertext: It first computes the FHE ciphertexts of all C_i's using the public key of the FHE scheme as

$$\mathsf{HEnc}(C) = (\mathsf{HEnc}(C_0), \mathsf{HEnc}(C_1), \cdots, \mathsf{HEnc}(C_{\ell-1})).$$

Step 2: Homomorphically evaluate the permutation circuit $\mathcal{C}_\pi$: Using the encrypted key $\mathsf{HEnc}(K\|N)$ of the AEnc scheme, it evaluates the permutation circuit $\mathcal{C}_\pi$ sequentially for each ciphertext block and obtain

$$\mathsf{HEnc}(K_i) \leftarrow \lfloor \mathsf{HEnc}\big(\pi^i(S)\big) \rfloor_r$$

and then computes the FHE ciphertext of M_i from $\mathsf{HEnc}(K_i)$ and $\mathsf{HEnc}(C_i)$ as

$$\mathsf{HEnc}(M_i) = \mathsf{HEnc}(K_i \oplus C_i) \leftarrow \mathsf{HEval}(\mathrm{XOR}, \mathsf{HEnc}(K_i), \mathsf{HEnc}(C_i)).$$

In the above steps, the most expensive operation is the homomorphic evaluation of the permutation. As we have considered the binary circuit, the choice of the FHE scheme determines the ciphertext $\mathsf{HEnc}(C_i)$ whether it is a single ciphertext (packed using SIMD) or r ciphertexts where each ciphertext is an FHE encryption of one-bit of AE ciphertexts.

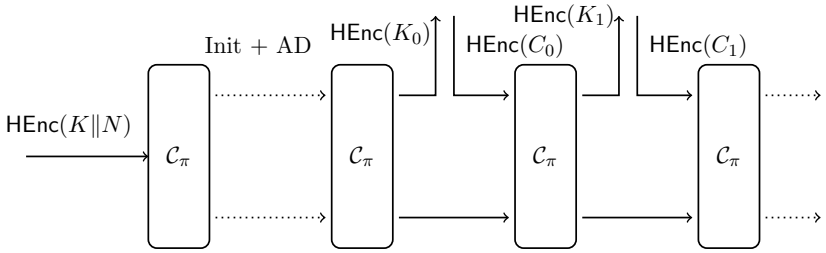

Figure 2: Homomorphic evaluation of the sponge mode in the decryption phase. Two blocks of decryption are shown where $\mathcal{C}_\pi$ is the circuit for the permutation π.

Note that for the permutation-based AE schemes such as ACE and ASCON, the same underlying permutation is required to evaluate for both encryption and decryption operations. Figure 2 presents a high-level description of the homomorphic evaluation of the sponge mode (without subtle details). On the other hand, for block ciphers based AE schemes, the decryption algorithm may have a different circuit from an encryption one depending upon the scheme. Note that to convert an AE ciphertext to an FHE ciphertext to enable homomorphic computation, the decryption circuit needs to be evaluated. In the following section, we focus only on the homomorphic evaluation of the core primitives of the AE schemes.

5.2 Experimental Evaluation

Experimental setup. We have developed a generic implementation of homomorphic evaluation of core-AE circuits in `C++` on top of the TFHE scheme [CGGI16a, CGGI16b], which is a candidate in the HE standard [ACC+18]. The TFHE supports homomorphic evaluations

of binary gates and does not required to know the depth of the circuit during the parameter generation phase. As the circuit is represented using only XOR, AND and NOT gates, our implementation uses homomorphic computations of only these three gates. In our implementation, we feed the core circuit of an AE scheme and an encrypted state of the primitive, and obtain the encrypted output through its homomorphic evaluation. For instance, for a permutation, we provide the FHE encrypted key and nonce, and homomorphically evaluate the permutation circuit and obtain the encrypted output. We use the default parameters of TFHE providing 110-bit security. We conduct the experiments on a desktop with a 3.00GHz Intel Core i7-9700 CPU and 32 GB RAM running on Ubuntu 18.04. Note that the homomorphic evaluation is done using a single thread (no parallelism is exploited).

Performance. We now present the Wall-clock running time for homomorphically evaluating the core-AE circuits in Table 2. We micro-benchmark the timings of homomorphic computation of XOR, AND and NOT gates in TFHE in Table 5. For instance, in TFHE, the homomorphic XOR computation time for 128 ciphertexts is about 4.96 seconds. The micro-benchmarking results show that the homomorphic operations for XOR and AND takes almost the same amount of time. Table 4 reports the computation time (in second) of the homomorphic evaluation of core AE circuits given in Table 2.

Table 5: Benchmarking homomorphic XOR, AND and NOT operations in TFHE on our machine. Time is given in second (s).

	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

Estimating time for individual modes. Note that Table 4 presents the homomorphic evaluation time only for the core primitives of the AE schemes. The modes of operation for different ciphers are different. Also, different AE schemes process different number of message blocks in each call of a core primitive. The homomorphic evaluation time of a particular AE mode using TFHE can be estimated based on the number of invocations of the core primitive and the homomorphic operation cost of the mode using Table 5.

6 Conclusions and Future Work

In this work we considered secure evaluation of lightweight authenticated ciphers for privacy-enhancing cryptographic applications and presented some preliminary results from our ongoing work. To our knowledge, this work is the first that reports the Boolean circuits of the core primitives of NIST lightweight cryptography round 2 candidates. Two implementations of secure 2PC evaluation and homomorphic evaluation of the lightweight AE schemes using existing garbled circuit and FHE libraries are developed. We presented the performance results for both

privacy-preserving evaluations. We are currently working on specialized secure computation and homomorphic evaluation techniques for lightweight ciphers and also applying for blockchain applications.

References

[AABS+19] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. Cryptology ePrint Archive, Report 2019/426, 2019. `https://eprint.iacr.org/2019/426`.

[ACC+18] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[AD18] Tomer Ashur and Siemen Dhooghe. Marvellous: a stark-friendly family of cryptographic primitives. Cryptology ePrint Archive, Report 2018/1098, 2018. `https://eprint.iacr.org/2018/1098`.

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for mpc, and more. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *Computer Security – ESORICS 2019*, pages 151–171, Cham, 2019. Springer International Publishing.

[AGR+16] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 191–219, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 2087?2104, New York, NY, USA, 2017. Association for Computing Machinery.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for mpc and fhe. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 430–454, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[BCC+19] Lawrence Bassham, Cagdas Calik, Donghoon Chang, Jinkeon Kang, Kerry McKay, and Meltem Sonmez Turan. Lightweight cryptography: Round 2 candidates, 2019. `https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates`.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communi-*

cations Security, CCS '12, page 784?796, New York, NY, USA, 2012. Association for Computing Machinery.

[BKL+07] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, pages 450–466, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[bri] Bristol fashion mpc circuits. `https://homes.esat.kuleuven.be/~nsmart/MPC/old-circuits.html`.

[BSS+15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The simon and speck lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference*, DAC '15, New York, NY, USA, 2015. Association for Computing Machinery.

[CAE] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. `https://competitions.cr.yp.to/caesar.html`.

[CCF+18] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. *Journal of Cryptology*, 31(3):885–916, 2018.

[CGGI16a] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 3–33, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[CGGI16b] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption library, August 2016. https://tfhe.github.io/tfhe/.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A cipher with low anddepth and few ands per bit. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 662–692, Cham, 2018. Springer International Publishing.

[DK10] Ivan Damgård and Marcel Keller. Secure multiparty aes. In Radu Sion, editor, *Financial Cryptography and Data Security*, pages 367–374, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[DKL+12] Ivan Damgård, Marcel Keller, Enrique Larraia, Christian Miles, and Nigel P. Smart. Implementing aes via an actively/covertly secure dishonest-majority mpc protocol. In Ivan Visconti and Roberto De Prisco, editors, *Security and Cryptography for Networks*, pages 241–263, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[ea18a] Martin Albrecht et al. Homomorphic encryption standards meeting 2018, 2018. `http://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf`.

[ea18b] Martin Albrecht et al. Homomorphic encryption standards meeting 2019, 2018. http://homomorphicencryption.org/aug-17-2019-homomorphicencryption-org-standards-meeting/.

[ECR] eSTREAM: the ecrypt stream cipher project. http://www.ecrypt.eu.org/stream/.

[FHK+14] Martin Franz, Andreas Holzer, Stefan Katzenbeisser, Christian Schallhart, and Helmut Veith. Cbmc-gc: An ansi c compiler for secure two-party computations. In Albert Cohen, editor, *Compiler Construction*, pages 244–249, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 626–645, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In *Proceedings of the 32nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, pages 850–867, Berlin, Heidelberg, 2012. Springer-Verlag.

[GKR+19] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. Cryptology ePrint Archive, Report 2019/458, 2019. https://eprint.iacr.org/2019/458.

[GRR+16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. Mpc-friendly symmetric key primitives. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS 16, page 430443, New York, NY, USA, 2016. Association for Computing Machinery.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols: Techniques and Constructions*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010.

[Hou17] R Housley. Using chacha20-poly1305 authenticated encryption in the cryptographic message syntax (cms). Technical report, RFC 8103, 2017.

[KMR12] Seny Kamara, Payman Mohassel, and Ben Riva. Salus: A system for server-aided secure function evaluation. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 797–808, New York, NY, USA, 2012. Association for Computing Machinery.

[KMS+16] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 839–858, 2016.

[KOR+17] Marcel Keller, Emmanuela Orsini, Dragos Rotaru, Peter Scholl, Eduardo Soria-Vazquez, and Srinivas Vivek. Faster secure multi-party computation of aes and des using lookup tables. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *Applied Cryptography and Network Security*, pages 229–249, Cham, 2017. Springer International Publishing.

[LN14] Tancrède Lepoint and Michael Naehrig. A comparison of the homomorphic encryption schemes fv and yashe. In David Pointcheval and Damien Vergnaud, editors, *Progress in Cryptology – AFRICACRYPT 2014*, pages 318–335, Cham, 2014. Springer International Publishing.

[LR15] Yehuda Lindell and Ben Riva. Blazing fast 2pc in the offline/online setting with security for malicious adversaries. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 579?590, New York, NY, USA, 2015. Association for Computing Machinery.

[LTW13] Sven Laur, Riivo Talviste, and Jan Willemson. From oblivious aes to efficient and secure database join in the multiparty setting. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*, pages 84–101, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Man20] Kalikinkar Mandal. Boolean circuits of lightweight authenticated ciphers, 2020. `http://www.cs.unb.ca/~kmandal/nistlwc/circuit.html`.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient fhe with low-noise ciphertexts. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 311–343, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 681–700, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[PSSW09] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 250–267, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[RSS17] Dragos Rotaru, Nigel P. Smart, and Martijn Stam. Modes of operation suitable for computing on encrypted data. *IACR Transactions on Symmetric Cryptology*, 2017(3):294–324, Sep. 2017.

[SCG+14] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[Sta19] ISO/IEC 29192-2:2019 Standards. Information security – lightweight cryptography – part 2: Block ciphers, 2019. `https://www.iso.org/standard/78477.html`.

[STMÇ+19] Meltem Sönmez Turan, Kerry McKay, Çağdaş Çalık, Donghoon Chang, and Lawrence Bassham. Status report on the first round of the nist lightweight cryptography standardization process. Technical report, National Institute of Standards and Technology, 2019.

[TS] Stefan Tillich and Nigel Smart. Aes circuit. `https://homes.esat.kuleuven.be/~nsmart/MPC/AES-non-expanded.txt`.

[WMK16] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit. `https://github.com/emp-toolkit`, 2016.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 220–250, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.