DK Lock: Dual Key Logic Locking Against Oracle-Guided Attacks

Jordan Maynard
Computer Engineering & Computer Science Department
California State University Long Beach
Long Beach, CA, USA
jordan.maynard@student.csulb.edu

Amin Rezaei
Computer Engineering & Computer Science Department
California State University Long Beach
Long Beach, CA, USA
amin.rezaei@csulb.edu

***Abstract*—The semiconductor industry must deal with different hardware threats like piracy and overproduction as a result of outsourcing manufacturing. While there are many proposals to lock the circuit using a global protected key only known to the designer, there exist numerous oracle-guided attacks that can examine the locked netlist with the assistance of an activated IC and extract the correct key. In this paper, by adopting a low-overhead structural method, we propose *DK Lock*, a novel Dual Key locking method that securely protects sequential circuits with two different keys that are applied to one set of key inputs at different times. DK Lock structurally adds an activation phase to the sequential circuit, and a correct key must be applied for several cycles to exit this phase. Once the circuit has been successfully activated, a new functional key must be applied to the same set of inputs to resume normal operation. DK Lock opens up new avenues for hardware IP protection by simultaneously refuting the single static key assumption of the existing attacks and overcoming the state explosion problem of state-of-the-art sequential logic locking methods. Our experiments confirm that DK Lock maintains a high degree of security with reasonable power and area overheads.**

***Index Terms*—Logic Locking, Logic Encryption, SAT Attack, Logic Obfuscation, Sequential Circuits, Dynamic Key**

I. Introduction and Background

The semiconductor chip business has numerous security concerns, including piracy and overproduction, because every Integrated Circuit (IC) often consists of several pieces from different offshore foundries, each with its own set of potential hardware hazards. Logic locking (a.k.a. logic encryption) technique [1], which adds extra key inputs to a given netlist, is a well-studied but yet highly fragile approach to preventing unauthorized ICs from working. After the manufactured ICs return from the foundry, the correct key, which is known only to the designer, must be placed into a tamper-proof memory to make the locked circuit operational.

Traditional logic locking schemes such as XOR-based [1] and MUX-based locking [2] were found to be susceptible to the SAT attack [3] that can report the correct key in a short time with the help of an activated IC. This led to the development of several SAT-resilient locking schemes for both combinational and sequential circuits [4]–[36]. Despite the range of methods used to uncover the original functionality and/or the correct key of a locked circuit, all the existing attacks [37]–[52] have one thing in common: *they only search for a single globally correct key.* While SLED [35] uses dynamic key values that change upon a chosen event in the circuit, it still utilizes a single static set of values to activate the correct key sequence and thus remains susceptible to existing attacks. Generally speaking, any approach that employs an initial "seed" can be vulnerable to state-of-the-art attacks because the seed functions as a global key that, if discovered by the attacker, can unlock the circuit easily and nullify the security.

Furthermore, in contrast to the commonality of sequential circuits in the market, there are a relatively small amount of sequential logic locking methods [32]–[36]. Several existing sequential locking schemes use Finite State Machine (FSM)-based approaches to hide IC functionality. HARPOON [32] is a proposed scheme that adds extra states to the original state transition graph which act as an authentication mode requiring a correct sequence of inputs to reach the original initial state. Active hardware metering [33] is another FSM-based approach that adds "black hole" states which lock the circuit in an inescapable state after the incorrect input sequence is applied. JANUS-HD [34] uses a set of coherent synthesis augmentations to achieve simultaneously high output corruptibility and pruning attack resilience. All of these methods have been deciphered by sequential attacks like KC2 [52] and RANE [50].

In addition, FSM locking uses a behavioral approach, meaning that the outputs are computed as a function of the input sequence. Adding authentication states using the behavioral model leads to the state explosion problem. Our solution is to use a structural approach instead of a behavioral one to add locking functionality. Instead of adding states to the FSM, leading to an abundance of new states that must be considered, our activation logic is added structurally. This allows activation to be achieved solely through key inputs without having to worry about primary input sequences.

Our proposed logic locking method structurally adds an activation phase in a given sequential circuit, and a correct initial key must be applied for several cycles to exit this phase. Once the circuit has been successfully "activated," a new final key must be applied to the same set of inputs for the circuit to resume normal operation. To the best of our knowledge, no locking scheme has been proposed with a fixed-size set of key inputs that require the application of multiple distinct sets of

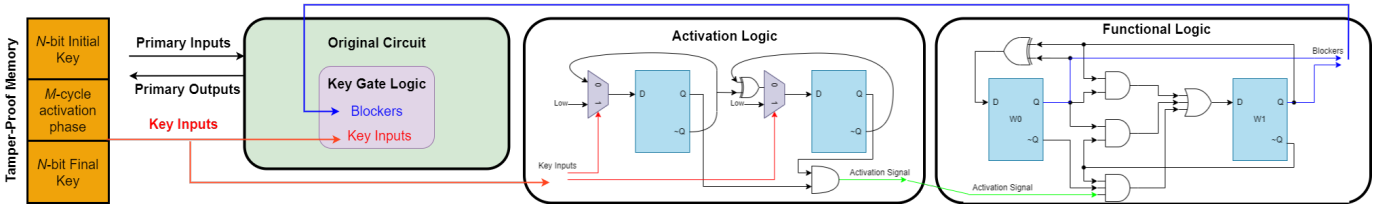
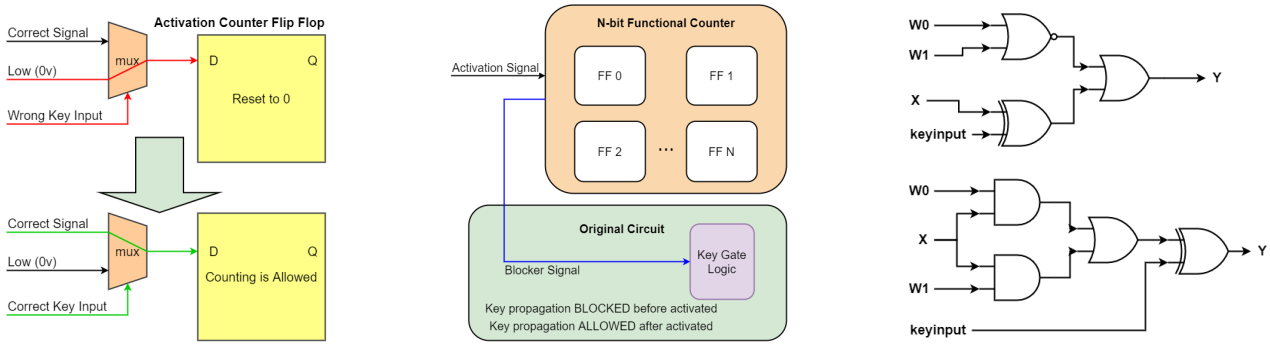


Fig. 1. DK Lock overview



(a) (b) (c)

Fig. 2. DK Lock structural modules (a) Activation counter FF (b) Functional logic (c) Integration examples

values to resume original functionality. This refutes the current assumption that finding just one correct key will uncover the original circuit functionality.

The state-of-the-art logic locking methods [4], [5], [7], [8] operate on the assumption that the attacker has complete access to the locked netlist. Furthermore, he/she can purchase a functioning circuit from the market as an oracle and get the correct outputs for given input vectors. Also, because practically all ICs are sequential circuits, it is presumed the attacker has access to the scan chain. This commonly known attacker paradigm is taken into account in this research. Despite there currently being no other dynamic key locks on a static set of key inputs, we still assume that the attacker is aware of the presence of two keys.

The main contributions of this paper are as follows:

- Proposing a novel logic locking scheme called Dual Key (DK) Lock that refutes the single static global key assumption of existing attacks.
- Overcoming the state explosion problem of state-of-the-art FSM locking by following a structural approach instead of an impractical behavioral one.
- Presenting the high security gain of the DK Lock against existing attacks with low power and area overhead.

II. Dual Key Lock

The main idea behind our proposed DK Lock is to use one set of key inputs for two different keys during different phases of operation. The activation phase can be characterized as a sequential lock, while the functional phase uses combinational locking. These two phases operate using the same set of key inputs, and they interact in a way that ensures the combinational portion cannot be interacted with until the sequential portion has been unlocked. An initial (i.e., activation) key must be applied to the locked circuit for a chosen number of cycles to "activate" the circuit. The circuit becomes activated and moves into the functional phase when the activation logic triggers a signal which changes the state of the functional logic. When in the functional phase, a final (i.e., functional) key must be applied to regain original circuit functionality. DK Lock is implemented by the addition of three structural modules into the circuit: activation logic, functional logic, and integration logic. Fig. 1 depicts a high-level overview of the interaction between the added lock and the original circuit.

A. Activation Phase

The activation logic is comprised of a key-controlled up-counter with variable bit size and logic that determines when the activation signal is triggered. The counter begins at "0" and counts up when the correct key is applied. Once m number of cycles is reached, the activation signal is triggered and the circuit enters the functional phase. The user is able to customize this logic by choosing the value m and the corresponding number of bits for the counter. For example, if m=9, and the counter size is four bits, then the circuit will activate when the counter bits hold the binary value 1001.

At each Flip Flop (FF) input in the activation counter, 2-1 MUXs are added to allow key-controlled counting. Each MUX main inputs are the correct counter signal and a low (i.e., "0"v) signal, and a key bit drives the selector. When an incorrect key bit is applied to the MUX of a certain FF, a low signal is fed to that FF input. Thus, the counter will count up only when the correct key is applied. Fig. 2a provides a low-level view of an activation counter FF. Based on the correct key value, the

Algorithm 1: DK Lock algorithm

Input: Original netlist $f(x)$, key size N, and # of cycles M
Output: Locked netlsit $g(x,k)$ and oracle $h(x)$
$k_a^* \leftarrow$ CreateCorrectKey(N);
// k_a^* is initial key
$k_f^* \leftarrow$ CreateCorrectKey(N);
// k_f^* is final key
$g(x,k) \leftarrow f(x)$;
$i \leftarrow 0$;
while *($i < N$)* **do**
  $activation\_flipflop \leftarrow$ NewActivationFF($k_{a_i}^*$,M);
  Add($activation\_flipflop$) to $activation\_logic$;
  $rand\_gate \leftarrow$ RandomGate($g(x,k)$);
  $key\_logic \leftarrow$ NewKeyGate($k_{f_i}^*$);
  Add(key_logic) to $g(x,k)$ at $rand\_gate$;
  $i \leftarrow i+1$;
Add($activation\_logic$) to $g(x,k)$;
Add($functional\_logic$) to $g(x,k)$;
$h(x) \leftarrow g(x,k)$;
for *(all key bits $k_i \in k$)* **do**
  Remove(key_input_i) from h(x);
  $fixed\_key\_mux \leftarrow$ *NewFixedKeyMux($k_{a_i}^*, k_{f_i}^*$);*
  Add($fixed\_key\_mux$) to h(x);
  $key\_gate_i \leftarrow fixed\_key\_mux$;
return $g(x,k), h(x)$;

MUX can be simplified to the AND of the key bit (or inverse of the key bit) with the correct signal.

When a correct activation key is applied for the specified *m* cycles, the activation counter's FFs will hold the binary value of *m*. The output of every FF is connected to a set of gates which output an activation signal to the functional logic once the binary value *m* is represented. This event triggers the activation of the circuit and begins the second phase of operation. A design consideration is that the value of *m* is not stored in any internal memory as it is built into the customized locking logic.

B. Functional Phase

The functional stage is characterized by the removal of blocker signals to allow key gate propagation in the circuit. A blocker is a signal which, when applied to a certain gate, causes a fixed output (i.e., a stuck-at fault) regardless of other gate inputs. Prior to circuit activation, each traditional key gate inserted into the original circuit has additional gates at its output. This added logic disallows the propagation of the key gates and thus the correct signals to the rest of the circuit during the activation phase. The functional logic interacts with the circuit integration logic (i.e., key gates) and removes the blockers once the functional stage is reached.

As shown in Fig. 2b, the functional logic consists of a modified *n*-bit ring counter which does not change state until activated. This counter is designed with low overhead, utilizing only *n* additional FFs. Once activated, the counter will continuously cycle through FF values without returning to the initial value set. Table I shows the states of a 2-bit functional counter, as an example. These FFs are connected to the blockers in key gate logic ($W0$ and $W1$), which is integrated with the original circuit at randomly chosen places (see Fig. 2c). Every state after activation allows propagation of traditional key gates, meaning the functional phase is equivalent to traditional single-gate locking. One design consideration is that the attacker does not know the initial state of the functional counter. In the 2-bit functional counter of Table I, "00" is the initial value set, but it can be any state for different designs.

TABLE I
2-BIT FUNCTIONAL COUNTER STATES

Activation Phase		Functional Phase	
Current	Next	Current	Next
00	00	00	01
00	00	01	10
00	00	10	11
00	00	11	01

C. Integration With Original Circuit

As previously mentioned, the activation phase of operation is completed by removing blockers from key gates. Two examples with two blocker signals are shown in Fig. 2c. A random signal from the original circuit *X* is chosen and the depicted logic is added. $W0$ and $W1$ will act as blockers during the activation phase, so both will have a logic low before activation. This causes corruption at the output *Y* regardless of the key input signal. During the functional phase, *Y* will equal *X* only if the correct final key bit is applied.

The presence of these blockers disallows observation of final key propagation to primary outputs of the circuit during the activation phase. This is the facet of DK Lock which preserves security of the final key while the initial key is being applied. SAT-based techniques are unable to resolve both keys because of their sequential application to the same set of inputs. The transition from activation to functional phase allows for final key propagation to primary outputs, but SAT-based attacks alone are unable to reach this stage without returning UNSAT model or an incorrect key.

In summary, activation and functional modules are connected to the original circuit through randomly inserted key logic. The addition of this logic along with a single set of dynamic key inputs effectively locks the circuit from all the state-of-the-art attacks that assume the existence of a single correct key. The correct operation of the circuit will be delayed until an initial key is applied for *m* cycles. Once the circuit is past activation, a different final key must be applied to the same set of key inputs to recover proper functionality. Algorithm 1 shows the pseudo-code of DK Lock.

TABLE II
COMBINATIONAL ATTACKS ON THE LOCKED BENCHMARKS WITH DK LOCK

Benchmark			Attacks		
Set	Benchmark	Key Size	SAT [3]	CycSAT [37]	Double-DIP [40]
Set 1	s27, s298, s349, s444, s510, s641, s713, s832, s953, s1196, s1488, s5378, s9234, s13207, s15850	Fixed Key Size of 10	UNSAT	UNSAT	UNSAT
Set 2	s27, s298, s349, s444, s510, s641, s713, s832, s953, s1196, s1488, s5378, s9234, s13207, s15850	6, 10, 21, 10, 27, 58, 59, 38, 40, 29, 28, 84, 13, 117, 98	UNSAT	UNSAT	UNSAT
Set 3	b01, b02, b03, b04, b06, b07, b08, b09, b10, b11, b12, b13, b14, b15, b17	Fixed Key Size of 10	UNSAT	UNSAT	UNSAT
Set 4	b01, b02, b03, b04, b06, b07, b08, b09, b10, b11, b12, b13, b14, b15, b17	2, 2, 4, 11, 2, 2, 2, 9, 2, 11, 7, 5, 10, 32, 36, 37	UNSAT	UNSAT	UNSAT

D. Obfuscation

One important security feature of our design is that the number of cycles *m* is hidden from the attacker. Instead of being stored in memory or somewhere the attacker may be able to access, it is built in to the logic itself. This deters SAT-based approaches further, adding many possibilities for the attacker to consider.

The dynamic nature of the unblocking signals from functional logic assist in obscuring key observability. Fixed state FFs in an activation-based design are an obvious flag to attackers, thus the addition of this characteristic to our structural design. The changing states of the functional logic also open flexibility for the design of key gate logic. The functionality of the designs shown in Fig. 2c may be realized through any number of unique layouts. By having several different approaches to key-gate logic within the design, structural analysis and removal attacks become more tedious for the attacker. Varying the gate-level implementation of these facets removes repetition within the design and hinders the attacker's ability to identify added logic by targeting repetition.

The functional counter logic is also more secure when the size is expanded (i.e., more FFs). This allows for dynamic states before and after activation, creating even further difficulty for observation by the attacker. Additional obfuscation currently implemented is a re-synthesis of the design which mixes the locking logic into the original circuit. However, the most beneficial approach would be a high-level transformation of the data-flow with the objective of making the gate-level design confusing to attackers. Identifying and tampering with critical signals in the design would be infeasible due to the obscurity of the logic. As a result, bypassing any part of the locking would prove significantly more difficult.

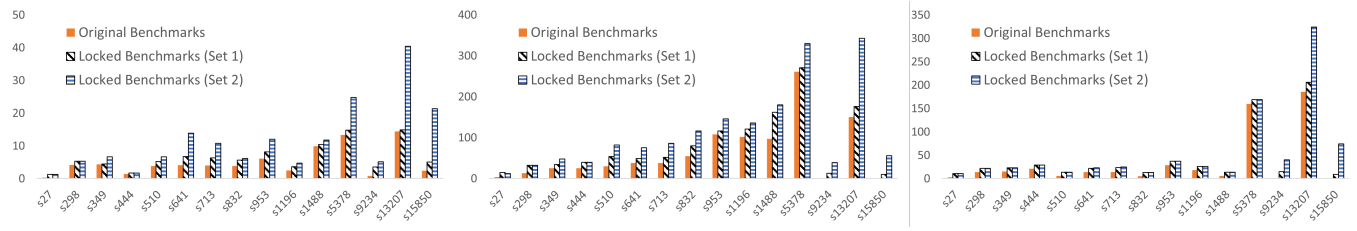
TABLE III
SEQUENTIAL ATTACKS ON THE LOCKED BENCHMARKS WITH DK LOCK - SET 1 & SET 3 (FIXED KEY SIZE OF 10)

Bench	KC2 [52]			RANE [50]	
	Time (s)	#Iter.	Result	Time (s)	Result
s27	0.82	99	Wrong Key	0.38	Wrong Key
s298	2.26	95	Wrong Key	0.92	Wrong Key
s349	-	2	Attack Failed	1.95	Wrong Key
s444	4.38	94	Wrong Key	0.97	Wrong Key
s510	140.04	96	Wrong Key	0.98	Wrong Key
s641	0.11	4	Wrong Key	2.39	Wrong Key
s713	133.03	96	Wrong Key	2.51	Wrong Key
s832	0.15	5	Wrong Key	1.22	Attack Failed
s953	-	1	Attack Failed	3.00	Wrong Key
s1196	0.13	2	Wrong Key	3.39	Wrong Key
s1488	-	2	Attack Failed	3.59	Wrong Key
s5378	-	2	Attack Failed	11.77	Wrong Key
s9234	2.81	6	Wrong Key	2.09	Attack Failed
s13207	485.63	98	Wrong Key	5.13	Attack Failed
s15850	637.85	98	Wrong Key	5.01	Attack Failed
b01	0.50	48	Wrong Key	0.79	Wrong Key
b02	0.39	50	Wrong Key	0.75	Wrong Key
b03	1.06	49	Wrong Key	0.88	Wrong Key
b04	-	49	Attack Failed	1.65	Attack Failed
b06	-	48	Attack Failed	0.77	Wrong Key
b07	3.27	49	Wrong Key	1.19	Attack Failed
b08	-	49	Attack Failed	0.9	Wrong Key
b09	5.07	50	Wrong Key	0.86	Wrong Key
b10	-	49	Attack Failed	0.93	Wrong Key
b11	600	38	Wrong Key	1.51	Wrong Key
b12	-	49	Attack Failed	1.92	Wrong Key
b13	-	49	Attack Failed	0.57	Attack Failed
b14	600	5	Wrong Key	11.21	Attack Failed
b15	600	17	Wrong Key	13.27	Attack Failed
b17	600	14	Wrong Key	51	Attack Failed

III. EXPERIMENTAL RESULTS

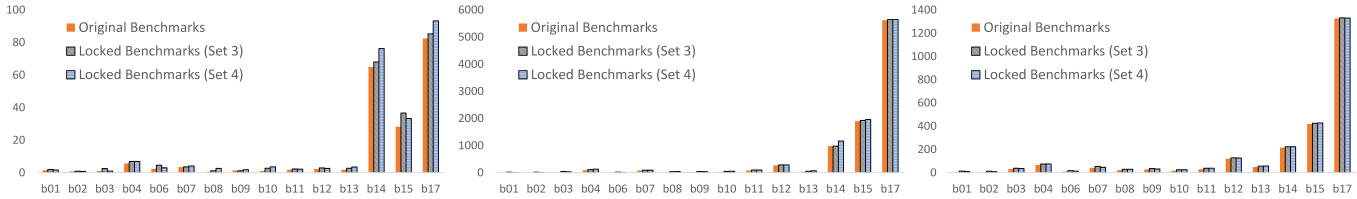
In this section, the robustness of DK Lock against state-of-the-art attacks is demonstrated on the ISCAS '89 [53] and ITC '99 [54] sequential benchmarks. All attacks are run in Ubuntu with 4GB of RAM. Please note that DK Lock is independent of the benchmark structure and thus applies to any circuit. Overall, four sets of benchmarks are created. The key size of $Set1$ & $Set3$ is fixed at 10, and the key size of $Set2$ & $Set4$ scales with the primary input size. The benchmarks are also implemented on the Nexys A7-100T FPGA board for area and power analysis. To implement the benchmarks on the FPGA, we use the ABC tool [55] to convert $.BENCH$ files to $.V$ files, followed by synthesis and implementation in Xilinx Vivado 2016.4 webpack edition.

Both combinational and sequential attacks were run against each benchmark to provide a wide array of results. The original SAT attack [3], CycSAT [37], and Double-DIP [40] were chosen for combinational attacks. For each locked benchmark, a combinational version was created by replacing FFs with scan inputs and outputs. Each FF input is replaced with a



(a) (b) (c)

Fig. 3. Overhead - ISCAS '89 benchmarks (a) Power Consumption (Watts) (b) Number of LUTs (c) Number of FFs



(a) (b) (c)

Fig. 4. Overhead - ITC '99 benchmarks (a) Power Consumption (Watts) (b) Number of LUTs (c) Number of FFs

primary output and each FF output is replaced with a primary input. The sequential attacks we ran against our locked circuits include KC2 [52] and RANE [50].

Table II shows the results of combinational attacks run on locked benchmarks with DK Lock. As anticipated, none of the existing attacks are able to find the correct key. The SAT-based combinational attacks report "UNSAT" since there is no single correct key that satisfies the equivalence of the locked circuit and the activated IC. The only case where they can report the correct key is when we use the same key for activation and functional phases, which reduces DK Lock to a single global key model. The key size has no effect on the strength of the proposed locking method against combinational attacks, as the single-key assumption is thwarted regardless of this factor.

Consider the way the SAT attack functions. Comparing the locked circuit to an oracle, the attack assumes that there is a single key that will make the functionality equivalent. As soon as a different output pattern is detected for some input pattern, the key is pruned. Since our lock requires two distinct keys, there is no single correct key that will return the circuit to normal functionality. This means that the locked circuit and oracle will continue to have different outputs for the same inputs until the circuit is unlocked. For this reason, the SAT attack will prune the correct activation key and never get a chance to consider the functional key.

Running the sequential attacks against DK Lock-protected benchmarks yielded similar results. Neither RANE [50] nor KC2 [52] were able to report correct keys for any of the locked benchmarks. Table III shows the results of both sequential attacks run against $Set1$ and $Set3$ benchmarks. Akin to the combinational attacks, the single global key assumption held true for the sequential attacks. KC2 reported only a single incorrect key when it was able to run through all solver iterations without preemptively terminating. RANE [50] was able to detect the presence of two keys; when a key was reported, it was split into two incorrectly sized wrong keys. As seen in Table III, considerably high run times were reported for KC2 [52] against benchmarks s510 and s15850. These further prove the security and scalability of DK Lock to large circuit sizes. Only a 10-bit key was used for all of these benchmarks, while a slight area and power overhead was incurred. In several smaller benchmarks, the iteration count of KC2 [52] reached near its set limit of 100.

Some benchmarks run against KC2 [52] proved incompatible with the attack for unknown reasons. In fact, multiple of the benchmarks which we now have results for did not run successfully at first. Take s13207 run against KC2 [52] for example. The first run was instantly aborted without providing any further information about why it failed. However, when the same original benchmark was locked with a different correct key set of the same size, the attack ran until the iteration limit and returned an incorrect key. Another example of this came from s27 locked with a 10 bit key and a 13 bit key. The 13-bit key version encountered an error during the run, but the 10-bit key locked benchmark returned a wrong key after reaching the iteration limit. This solidifies the validity of our results and even suggests that certain conditions may make the locked benchmark more secure against current attacks.

Several of the benchmarks run against RANE [50] also seemed to give issues related to the attack framework itself. At first, none of our locked benchmarks seemed to be compatible with the attack. The reason for these problems was we applied DK Lock to $.BENCH$ benchmarks, while RANE is compatible with only the $.V$ file format. The RANE attack uses

a different $.BENCH$ to $.V$ conversion tool than we first used to parse and rewrite our locked circuits. After determining this as the initial source of error, the built-in converter was used instead. This allowed the attack to report the results shown in Table III. For all remaining benchmarks which still caused the attack to report errors, these 10-bit key locked circuits were recreated from scratch using our script. These new circuits locked with different randomized keys were then converted to $.V$ format using the provided tool in RANE [50] and the attack was re-run against them. The benchmark s13207 was the only one which newly became compatible with the attack after these measures were taken. While it is possible that different versions of these problematic benchmarks with varied key sizes could prove compatible, time constraints prevented a full exploration into this. Regardless of the error status of some cases, not a single correct key was reported by the attack for any of the given benchmarks.

Fig. 3 and Fig. 4 show the power usage of the locked benchmarks of each set compared with the original ones, as well as the resource utilization (i.e., number of LUTs and FFs) of the original and locked benchmarks implemented on the Nexys A7-100T FPGA board. As can be seen, DK Lock scales well in terms of overhead. For example, the power consumption increase of $s13207$ is only 3% with 17% of more LUTs and 11% of more FFs under a key size of 10 ($Set1$). The power and resource utilization are obviously higher when we use larger key sizes, but as the attack evaluation findings verified, we can safely lock any circuit with a fixed key size that is independent of the circuit size. For instance, there is no point in locking $s713$ with a key size of 59 ($Set2$) to face double power consumption and resource utilization while it can gain the same security benefit with a much smaller key size ($Set1$).

IV. Conclusion & Future Outlook

In this paper, we introduced the idea of multiple key values being inserted into the fixed-size set of key inputs. Formally, we proposed a dual key logic locking method named DK Lock that structurally adds an activation phase in any given sequential circuit which requires an activation key to be applied for several cycles. After this activation phase is completed, another key must be applied to the same set of key inputs to move into normal operation. Experimental results showed that DK Lock is secure against state-of-the-art attacks [3], [37], [40], [50], [52] on logic locking schemes while consuming reasonable power and area overheads.

DK Lock opens up a new avenue for sequential defenses against IC piracy and overproduction. Our scheme introduces a significant weakness of state-of-the-art combinational and sequential attacks by voiding the single static global key assumption. With multiple keys needed to unlock the circuit, current attacks are stumped by their search for a single correct key. As we predicted and as our experimental results demonstrate, dynamic key inputs provide strong resistance to attacks on both sequential and combinational benchmarks. Based on these findings, multi-key schemes is a promising direction for further research in sequential logic locking.

An expansion of dual key locking to include any number of keys is a promising path for future works. This could take shape in a multi-level activation phase which unlocks different parts of the original circuit at each stage. Another separate approach could consist of changing the correct final key on several fixed-cycle periods. If implemented using a structural approach, these designs may prove to add a high level of protection without severely impacting area and power overheads.

Acknowledgment

This work is supported by the National Science Foundation under Award No. 2245247.

References

[1] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1069-1074, 2008.

[2] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," In *Design Automation Conference (DAC)*, pp. 83–89, 2012.

[3] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," In *International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137-143, 2015.

[4] M. Yasin, B. Mazumdar, J. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," In *International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236-241, 2016.

[5] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," In *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 127-146, 2016.

[6] Y. Shen, A. Rezaei, and H. Zhou, "A comparative investigation of approximate attacks on logic encryptions," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 271-276, 2018.

[7] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1601–1618, 2017.

[8] A. Rezaei, Y. Shen, and H. Zhou, "Rescuing logic encryption in post-SAT era by locking & obfuscation," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 13-18, 2020.

[9] H. Zhou, A. Rezaei, and Y. Shen, "Resolving the trilemma in logic encryption," In *International Conference on Computer Aided Design (ICCAD)*, pp. 1-8, 2019.

[10] A. Rezaei, Y. Shen, S. Kong, J. Gu and H. Zhou, "Cyclic locking and memristor-based obfuscation against CycSAT and inside foundry attacks," In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 85-90, 2018.

[11] A. Rezaei, Y. Li, Y. Shen, S. Kong, and H. Zhou, "CycSAT-unresolvable cyclic logic encryption using unreachable states" In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 358–363, 2019.

[12] A. Rezaei, J. Gu, and H. Zhou, "Hybrid memristor-CMOS obfuscation against untrusted foundries," In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 535-540, 2019.

[13] X. Yang, P. Chen, H. Chiang, C. Lin, Y. Chen, and C. Wang, "LOOPLock 2.0: An enhanced cyclic logic locking approach" In *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 41, no. 1, pp. 29–34, 2021.

[14] H. M. Kamali, K. Z. Azar, H. Homayoun, and A. Sasan, "Full-Lock: Hard distributions of SAT instances for obfuscating circuits using fully configurable logic and routing blocks," In *Proceedings of Design Automation Conference (DAC)*, pp. 1-6., 2019.

[15] K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "Cross-lock: Dense layout-level interconnect locking using cross-bar architectures," In *Proceedings of the on Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 147– 152, 2018.

[16] B. Hu, J. Tian, M. Shihab, G. Reddy, W. Swartz, Y. Makris, B. C. Schaefer, and C. Sechen, "Functional obfuscation of hardware accelerators through selective partial design extraction onto an embedded FPGA," In *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 171–176, 2019.

[17] P. Mohan, O. Atli, J. Sweeney, O. Kibar, L. Pileggi, and K. Mai, "Hardware redaction via designer-directed fine-grained eFPGA insertion," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1186–1191, 2021.

[18] J. Bhandari, A. Moosa, B. Tan, C. Pilato, G. Gore, X. Tang, S. Temple, P. Gaillardon, and R. Karri, "Exploring eFPGA-based redaction for IP protection," In *International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.

[19] Z. U. Abideen, T. D. Perez and S. Pagliarini, "From FPGAs to obfuscated eASICs: Design and security trade-offs," In *Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1-4, 2021

[20] R. Karmakar, H. Kumar, and S. Chattopadhyay, "Efficient key-gate placement and dynamic scan obfuscation towards robust logic encryption," In *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 2109-2124, 2019.

[21] U. Guin, Z. Zhou, and A. Singh, "Robust design-for-security architecture for enabling trust in IC manufacturing and test," In *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 5, pp. 818–830, 2018.

[22] G. Zhang, B. Li, B. Yu, D. Z. Pan, and U. Schlichtmann, "Timing camouflage: Improving circuit security against counterfeiting by unconventional timing," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 91–96, 2018.

[23] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against IC counterfeiting," In *Design Automation Conference (DAC)*, pp. 1–9, 2017.

[24] J. Sweeney, V. Zackriya, V S. Pagliarini, and L. Pileggi, "Latch-based logic locking," In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 132–141, 2020.

[25] A. Rezaei, A. Hedayatipour, H. Sayadi, M. Aliasgari, and H. Zhou, "Global attack and remedy on IC-specific logic encryption," In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 145-148, 2022.

[26] D. Sisejkovic, F. Merchant, L. M. Reimann, and R. Leupers, "Deceptive logic locking for hardware integrity protection against machine learning attacks" In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 6, pp. 1716-1729, 2022.

[27] R. Muttaki, R. Mohammadivojdan, M. Tehranipoor, and F. Farahmandi, "HLock: Locking IPs at the high-level language," In *Design Automation Conference (DAC)*, pp. 79–84, 2021.

[28] H. Zhou, Y. Shen, and A. Rezaei, "Vulnerability and remedy of stripped function logic locking," In *Cyptology ePrint Archive*, report 2019/139, 2019.

[29] N. Limaye, A. Chowdhury, C. Pilato, M. Nabeel, O. Sinanoglu, S. Garg, and R. Karri, "Fortifying RTL locking against oracle-Less (untrusted foundry) and oracle-guided attacks," In *Design Automation Conference (DAC)*, pp. 91–96, 2021.

[30] J. Slowik, G. Williams, R. Albashir, A. Samagio, G. S. Nicholas and F. Saqib, "Dynamic key updates for LUT locked design," In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 105-108, 2022.

[31] R. Afsharmazayejani, H. Sayadi, and A. Rezaei, "Distributed logic encryption: Essential security requirements and low-overhead implementation," In *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 127-131, 2022.

[32] R. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection." In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493-1502, 2009.

[33] Y. M. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security" In *USENIX Security Symposium on USENIX Security Symposium*, article 20, pp. 1–16, 2007.

[34] L. Li and A. Orailoglu, "JANUS-HD: Exploiting FSM sequentiality and synthesis flexibility in logic obfuscation to thwart SAT attack while offering strong corruption," In *Design, Automation & Test in Europe Conf. (DATE)*, pp. 1–6, 2022.

[35] Y. Kasarabada, V. Muralidharan, and R. Vemuri, "SLED: Sequential logic encryption using dynamic keys," In *International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 844-847, 2020.

[36] A. Rezaei and H. Zhou, "Sequential logic encryption against model checking attack," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1178-1181, 2021.

[37] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-based attack on cyclic logic encryptions," In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 49–56, 2017.

[38] N. Limaye, S. Patnaik, and O. Sinanoglu, "Fa-SAT: Fault-aided SAT-based attack on compound logic locking techniques," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1166-1171, 2021.

[39] Y. Shen, Y. Li, S. Kong, A. Rezaei, and H. Zhou, "SigAttack: New high-level SAT-based attack on logic encryptions," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 940-943, 2019.

[40] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," In *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 179-184, 2017.

[41] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," In *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517-532, 2020.

[42] Y. Shen, A. Rezaei, and H. Zhou, "SAT-based bit-flipping attack on logic encryptions," In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 629-632, 2018.

[43] D. Sirone and P. Subramanyan, "Functional analysis attacks on logic locking," In *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2514-2527, 2020.

[44] M. E. Massad, S. Garg, and M. Tripunitara, "Reverse engineering camouflaged sequential circuits without scan access," In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 33–40, 2017.

[45] Y. Kasarabada, S. Chen, and R. Vemuri, "On SAT-based attacks on encrypted sequential logic circuits," In *International Symposium on Quality Electronic Design (ISQED)*, pp. 204-211, 2019.

[46] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability assessment tool and attack for provably-secure logic locking techniques," In *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744-759, 2022.

[47] A. Saha and U. Chatterjee and D. Mukhopadhyay and R. S. Chakraborty, "DIP Learning on CAS-Lock: Using Distinguishing Input Patterns for Attacking Logic Locking" In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 688-693, 2022.

[48] Y. Shen, Y. Li, A. Rezaei, S. Kong, D. Dlott and H. Zhou, "BeSAT: Behavioral SAT-based attack on cyclic logic encryption," In *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp 657-662, 2019.

[49] A. Rezaei, R. Afsharmazayejani, and J. Maynard, "Evaluating the security of eFPGA-based redaction algorithms," In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, article 157, pp 1-7, 2022.

[50] S. Roshanisefat, H. M. Kamali, H. Homayoun, and A. Sasan, "RANE: An open-source formal de-obfuscation attack for reverse engineering of logic encrypted circuits," In *Great Lakes Symposium on VLSI (GLSVLSI)*, pp. 221–228, 2021.

[51] C. Karfa, R. Chouksey, C. Pilato, S. Garg, and R. Karri, "Is register transfer level locking secure?" In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 550–555, 2020.

[52] K. Shamsi, M. Li, D. Z. Pan and Y. Jin, "KC2: Key-condition crunching for fast sequential circuit deobfuscation" In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 534-539, 2019.

[53] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1929-1934, 1989.

[54] S. Davidson, "ITC'99 Benchmark Circuits - Preliminary Results," In *International Test Conference (ITC)*, pp. 1125-1125, 1999.

[55] Berkeley Logic Synthesis and Verification Group, "ABC: A system for sequential synthesis and verification," http://www.eecs.berkeley.edu/˜alanmi/abc/