A chosen plaintext attack on SILC and CLOC

Hassan Sadeghi, Javad Alizadeh

November 7, 2014

Abstract

SILC and CLOC are two submissions to the CAESAR competition with similar constructions. In this note we show SILC and CLOC don't provide IND-CPA.
Keywords: SILC, CLOC, IND-CPA.

1 Introduction

IND-CPA [1] security is a security definition for private- or public-key encryption schemes. At a high level, IND-CPA security means that no adversary can distinguish between encryptions of different messages, even when allowed to make encryptions on its own.

Definition 1.1. IND-CPA (for private-key encryption)
Let A be an adversary, which we model as an arbitrary non-uniform PPT machine (polynomial in the implicit security parameter of the encryption scheme). We define the following experiment/game played against A:
1. We (privately) choose a key K according to the key generation algorithm $K \leftarrow$ KeyGen
2. We (privately) choose a random bit $b \leftarrow \{0, 1\}$
3. Repeatedly do:
. A is allowed to query an oracle that computes the functionality Enc_K
. Challenge: A outputs two messages, M_0 and M_1
. Response: We give A the ciphertext Enc(M_b])
4. A outputs b' (i.e, a guess for our b)
We say that the advantage of A in this experiment is $Pr(b = b') - \frac{1}{2}$.

In this note we present an adversary for this experiment/game on SILC and CLOC that it's advantage is not negligible. First, for clarity of notation, we will describe SILC and CLOC by initial information.

2 SILC and CLOC

SILC [2] and CLOC [3] are two blockcipher modes of operation for authenticated encryption with associated data (AEAD). SILC is built upon CLOC cipher. SILC and ClOC

Algorithm SILC-$\mathcal{E}_K(N, A, M)$

1. $V \leftarrow \mathsf{HASH}_K(N, A)$
2. $C \leftarrow \mathsf{ENC}_K(V, M)$
3. $T \leftarrow \mathsf{PRF}_K(V, C)$
4. **return** (C, T)

Algorithm SILC-$\mathcal{D}_K(N, A, C, T)$

1. $V \leftarrow \mathsf{HASH}_K(N, A)$
2. $T^* \leftarrow \mathsf{PRF}_K(V, C)$
3. **if** $T \neq T^*$ **then return** $\perp$
4. $M \leftarrow \mathsf{DEC}_K(V, C)$
5. **return** M

Figure 1: Pseudocode of the encryption and the decryption algorithms of SILC

Algorithm $\mathsf{HASH}_K(N, A)$

1. $S_{\mathsf{H}}[0] \leftarrow E_K(\mathsf{zpp}(N))$
2. **if** $|A| = 0$ **then**
3. $\quad V \leftarrow \mathsf{g}(S_{\mathsf{H}}[0] \oplus \mathsf{Len}(A))$ // $\mathsf{Len}(A) = 0^n$
4. $\quad$ **return** V
5. $(A[1], \ldots, A[a]) \xleftarrow{n} A$
6. **for** $i \leftarrow 1$ **to** $a - 1$ **do**
7. $\quad S_{\mathsf{H}}[i] \leftarrow E_K(S_{\mathsf{H}}[i-1] \oplus A[i])$
8. $S_{\mathsf{H}}[a] \leftarrow E_K(S_{\mathsf{H}}[a-1] \oplus \mathsf{zap}(A[a]))$
9. $V \leftarrow \mathsf{g}(S_{\mathsf{H}}[a] \oplus \mathsf{Len}(A))$
10. **return** V

Algorithm $\mathsf{PRF}_K(V, C)$

1. $S_{\mathsf{P}}[0] \leftarrow E_K(\mathsf{g}(V))$
2. **if** $|C| = 0$ **then**
3. $\quad U \leftarrow \mathsf{g}(S_{\mathsf{P}}[0] \oplus \mathsf{Len}(C))$ // $\mathsf{Len}(C)$
4. $\quad T \leftarrow \mathsf{msb}_\tau(E_K(U))$
5. $\quad$ **return** T
6. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
7. **for** $i \leftarrow 1$ **to** $m - 1$ **do**
8. $\quad S_{\mathsf{P}}[i] \leftarrow E_K(S_{\mathsf{P}}[i-1] \oplus C[i])$
9. $S_{\mathsf{P}}[m] \leftarrow E_K(S_{\mathsf{P}}[m-1] \oplus \mathsf{zap}(C[m]))$
10. $U \leftarrow \mathsf{g}(S_{\mathsf{P}}[m] \oplus \mathsf{Len}(C))$
11. $T \leftarrow \mathsf{msb}_\tau(E_K(U))$
12. **return** T

Algorithm $\mathsf{ENC}_K(V, M)$

1. **if** $|M| = 0$ **then**
2. $\quad C \leftarrow \varepsilon$
3. $\quad$ **return** C
4. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
5. $S_{\mathsf{E}}[1] \leftarrow E_K(V)$
6. **for** $i \leftarrow 1$ **to** $m - 1$ **do**
7. $\quad C[i] \leftarrow S_{\mathsf{E}}[i] \oplus M[i]$
8. $\quad S_{\mathsf{E}}[i+1] \leftarrow E_K(\mathsf{fix1}(C[i]))$
9. $C[m] \leftarrow \mathsf{msb}_{|M[m]|}(S_{\mathsf{E}}[m]) \oplus M[m]$
10. $C \leftarrow (C[1], \ldots, C[m])$
11. **return** C

Algorithm $\mathsf{DEC}_K(V, C)$

1. **if** $|C| = 0$ **then**
2. $\quad M \leftarrow \varepsilon$
3. $\quad$ **return** M
4. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
5. $S_{\mathsf{D}}[1] \leftarrow E_K(V)$
6. **for** $i \leftarrow 1$ **to** $m - 1$ **do**
7. $\quad M[i] \leftarrow S_{\mathsf{D}}[i] \oplus C[i]$
8. $\quad S_{\mathsf{D}}[i+1] \leftarrow E_K(\mathsf{fix1}(C[i]))$
9. $M[m] \leftarrow \mathsf{msb}_{|C[m]|}(S_{\mathsf{D}}[m]) \oplus C[m]$
10. $M \leftarrow (M[1], \ldots, M[m])$
11. **return** M

Figure 2: Subroutines used in the encryption and decryption algorithms of SILC

Algorithm CLOC-$\mathcal{E}_K(N, A, M)$

1. $V \leftarrow \mathsf{HASH}_K(N, A)$
2. $C \leftarrow \mathsf{ENC}_K(V, M)$
3. $T \leftarrow \mathsf{PRF}_K(V, C)$
4. **return** (C, T)

Algorithm CLOC-$\mathcal{D}_K(N, A, C, T)$

1. $V \leftarrow \mathsf{HASH}_K(N, A)$
2. $T^* \leftarrow \mathsf{PRF}_K(V, C)$
3. **if** $T \neq T^*$ **then return** $\perp$
4. $M \leftarrow \mathsf{DEC}_K(V, C)$
5. **return** M

Figure 3: Pseudocode of the encryption and the decryption algorithms of CLOC

Algorithm $\mathsf{HASH}_K(N, A)$

1. $(A[1], \ldots, A[a]) \xleftarrow{n} A$
2. $S_{\mathsf{H}}[1] \leftarrow E_K(\mathsf{fix0}(\mathsf{ozp}(A[1])))$
3. **if** $\mathsf{msb}_1(\mathsf{ozp}(A[1])) = 1$ **then**
4. $\quad S_{\mathsf{H}}[1] \leftarrow \mathsf{h}(S_{\mathsf{H}}[1])$
5. **if** $a \geq 2$ **then**
6. $\quad$ **for** $i \leftarrow 2$ **to** $a-1$ **do**
7. $\quad\quad S_{\mathsf{H}}[i] \leftarrow E_K(S_{\mathsf{H}}[i-1] \oplus A[i])$
8. $\quad S_{\mathsf{H}}[a] \leftarrow E_K(S_{\mathsf{H}}[a-1] \oplus \mathsf{ozp}(A[a]))$
9. **if** $|A[a]| = n$ **then**
10. $\quad V \leftarrow \mathsf{f}_1(S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N))$
11. **else** // $0 \leq |A[a]| \leq n-1$
12. $\quad V \leftarrow \mathsf{f}_2(S_{\mathsf{H}}[a] \oplus \mathsf{ozp}(N))$
13. **return** V

Algorithm $\mathsf{PRF}_K(V, C)$

1. **if** $|C| = 0$ **then**
2. $\quad T \leftarrow \mathsf{msb}_\tau(E_K(\mathsf{g}_1(V)))$
3. $\quad$ **return** T
4. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
5. $S_{\mathsf{P}}[0] \leftarrow E_K(\mathsf{g}_2(V))$
6. **for** $i \leftarrow 1$ **to** $m-1$ **do**
7. $\quad S_{\mathsf{P}}[i] \leftarrow E_K(S_{\mathsf{P}}[i-1] \oplus C[i])$
8. **if** $|C[m]| = n$ **then**
9. $\quad S_{\mathsf{P}}[m] \leftarrow E_K(\mathsf{f}_1(S_{\mathsf{P}}[m-1] \oplus C[m]))$
10. **else** // $1 \leq |C[m]| \leq n-1$
11. $\quad S_{\mathsf{P}}[m] \leftarrow E_K(\mathsf{f}_2(S_{\mathsf{P}}[m-1] \oplus \mathsf{ozp}(C[m])))$
12. $T \leftarrow \mathsf{msb}_\tau(S_{\mathsf{P}}[m])$
13. **return** T

Algorithm $\mathsf{ENC}_K(V, M)$

1. **if** $|M| = 0$ **then**
2. $\quad C \leftarrow \varepsilon$
3. $\quad$ **return** C
4. $(M[1], \ldots, M[m]) \xleftarrow{n} M$
5. $S_{\mathsf{E}}[1] \leftarrow E_K(V)$
6. **for** $i \leftarrow 1$ **to** $m-1$ **do**
7. $\quad C[i] \leftarrow S_{\mathsf{E}}[i] \oplus M[i]$
8. $\quad S_{\mathsf{E}}[i+1] \leftarrow E_K(\mathsf{fix1}(C[i]))$
9. $C[m] \leftarrow \mathsf{msb}_{|M[m]|}(S_{\mathsf{E}}[m]) \oplus M[m]$
10. $C \leftarrow (C[1], \ldots, C[m])$
11. **return** C

Algorithm $\mathsf{DEC}_K(V, C)$

1. **if** $|C| = 0$ **then**
2. $\quad M \leftarrow \varepsilon$
3. $\quad$ **return** M
4. $(C[1], \ldots, C[m]) \xleftarrow{n} C$
5. $S_{\mathsf{D}}[1] \leftarrow E_K(V)$
6. **for** $i \leftarrow 1$ **to** $m-1$ **do**
7. $\quad M[i] \leftarrow S_{\mathsf{D}}[i] \oplus C[i]$
8. $\quad S_{\mathsf{D}}[i+1] \leftarrow E_K(\mathsf{fix1}(C[i]))$
9. $M[m] \leftarrow \mathsf{msb}_{|C[m]|}(S_{\mathsf{D}}[m]) \oplus C[m]$
10. $M \leftarrow (M[1], \ldots, M[m])$
11. **return** M

Figure 4: Subroutines used in the encryption and decryption algorithms of CLOC

take three parameters, a blockcipher E, a nonce length l_N and a tag length τ where l_N and τ in bits. Procedures of the encryption and the decryption of SILC and CLOC are explained in figures (1) and (3). In these algorithms, we use four subroutines, HASH, PRF, ENC, and DEC that are defined in in figures (2) and (4).In this not we use only function $fix1$ that is defined

$$fix1(X) := X \vee 10^{n-1}$$

3 chosen plaintext attack

In this section we present adversary B for experiment/game of IND-CPA as follow:
First adversary B outputs two massages $M_0 = M[1]||M[2]...||M[n]$ and $M_1 = \tilde{M}[1]||\tilde{M}[2]...||\tilde{M}[n]$ such that for all $1 < i, j < n$

$$M[i] \oplus M[j] \neq \tilde{M}[i] \oplus \tilde{M}[j], \quad M[i] \oplus M[j] \notin \{0^n, 10^{n-1}\}, \quad \tilde{M}[i] \oplus \tilde{M}[j] \notin \{0^n, 10^{n-1}\}.$$

We (privately) choose a key K, a nonce N, an associated data A and a random bit $b \leftarrow \{0, 1\}$ then we give adversary B the ciphertext $C = C[1]||C[2]||...C[n]$ where

$$(C, T) = SILC_\xi(N, A, M_b) \ \Big((C, T) = CLOC_\xi(N, A, M_b)\Big).$$

Adversary B examines C in three distinct cases:
Case1 : $\exists\ \mathbf{1 \le i, j \le (n-1)};\ \mathbf{C[i] = C[j]}\ \mathbf{or}\ \mathbf{C[i] \oplus C[j] = 10^{n-1}}$
In this case we have $fix1(C[i]) = fix1(C[j])$ so we have

$$C[i+1] \oplus C[j+1] = M[i+1] \oplus M[j+1]\ \ or\ \ C[i+1] \oplus C[j+1] = \tilde{M}[i+1] \oplus \tilde{M}[j+1]$$

If $C[i+1] \oplus C[j+1] = M[i+1] \oplus M[j+1]$ then M_0 is plaintext of C and adversary B outputs $b' = b = 0$ else M_1 is plaintext of C and B outputs $b' = b = 1$.
Case2 : $\exists\ \mathbf{2 \le i, j \le n};\ \mathbf{C[i] \oplus C[j] = M[i] \oplus M[j]}$
If case 1 did not occur then M_1 will be plaintext of C because if M_0 be plaintext of C by $C[i] \oplus C[j] = M[i] \oplus M[j]$ we conclude $S_E[i] = S_E[j]$ and $fix1(C[i-1]) = fix1(C[j-1])$, equivalently

$$C[i-1] = C[j-1],\ \ C[i-1] \oplus C[j-1] = 10^{n-1}$$

while we had assumed case1 does not befall so adversary B outputs $b' = b = 1$
Case3 : $\exists\ \mathbf{2 \le i, j \le n};\ \mathbf{C[i] \oplus C[j] = \tilde{M}[i] \oplus \tilde{M}[j]}$
By similar proving in case2 adversary finds M_0 is plaintext of C and outputs $b' = b = 0$.
Final operation:
If $Case1$,$Case2$ and $Case3$ did not occur adversary chooses a key $\hat{K}$ and puts $\hat{C}_m := E_{\hat{K}}(fix1(C[m]))$ and examines C in the following distinct two cases:
FirstCase : $\exists\ \mathbf{1 \le i \le n-1};\ \mathbf{\hat{C}[i] = C[i+1] \oplus M[i+1]}$
If M_0 be plaintext of C then by $\hat{C}[i] = C[i+1] \oplus M[i+1]$ we conclude $E_{\hat{K}}(fix1(C[i])) = E_K(fix1(C[i]))$ and $K = \hat{K}$ so for $j = 1$ to $n-1$ we must have

$$\hat{C}[j] = C[j+1] \oplus M[j+1] \tag{1}$$

If (1) did not occur then adversary outputs $b' = b = 1$.
SecondCase : $\exists\ \mathbf{1 \le i \le n-1};\ \mathbf{\hat{C}[i] = C[i+1] \oplus \tilde{M}[i+1]}$
If M_1 be plaintext of C then by $\hat{C}[i] = C[i+1] \oplus \tilde{M}[i+1]$ we conclude $E_{\hat{K}}(fix1(C[i])) = E_K(fix1(C[i]))$ and $K = \hat{K}$ so for $j = 1$ to $n-1$ we must have

$$\hat{C}[j] = C[j+1] \oplus \tilde{M}[j+1] \tag{2}$$

If (2) did not occur then adversary outputs $b' = b = 0$.
If first case and second case did not occurred adversary repeat final operation by choosing another key.
Now, we compute the advantage of adversary B. Probability of occurrence case1 is $2C(n-1,2)2^{-128}$ and probability of occurrence case2 and case3 is $2C(n-1,2)2^{-128}$ and if adversary repeat final operation in m times, probability of victory of adversary is $2m(n-1)2^{-128}$ so advantage of adversary B is

$$Adv(B) = pr(b = b') - \frac{1}{2} = 4C(n-1,2)2^{-128} + 2m(n-1)2^{-128}$$

where m is number of frequency of final operation.

4 conclusion

in this note we conclude SILC and CLOC are not indistinguishable against chosen plaintext attack and there exist adversaries can distinguish between encryptions of different messages.

References

[1] S. Goldwasser and S. Micali. Probabilistic encryption. Journal of Computer and System Sciences, 28:270-299, 1984.

[2] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka, Eita Kobayashi, SILC v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/round1/silcv1.pdf.

[3] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, Sumio Morioka, Eita Kobayashi, CLOC v1. CEASAR Cryptographic Competitions, 2014. http://competitions.cr.yp.to/round1/clocv1.pdf.

Hassan Sadeghi
Department of Mathematics, Faculty of Science
University of Qom
Qom. Iran
Email: sadeghihassan64@gmail.com