Security of Symmetric Encryption Schemes with One-Way IND-CNA Key Setup

Bartosz Zoltak

http://www.vmpcfunction.com
bzoltak@vmpcfunction.com

Abstract. We analyse the consequences of specific properties of the key-setup phase in symmetric encryption schemes for their security. We find that key-setup routines satisfying IND-CNA and one-wayness allow to construct schemes which are provably secure against key-recovery attacks. We propose a specific cryptosystem based on a stream cipher with a one-way IND-CNA key-setup, for which we present a proof, based on a set of scheme-specific assumptions, that it remains secure even if a successful key-recovery attack against the underlying cipher is found.

Keywords: Provable Security, Stream Cipher, Key Scheduling Algorithm

1 Introduction

There has been a lot of interest in proofs of security for cryptographic schemes in recent years. A commonly employed approach is reduction and basing the proof on an assumption or a set of assumptions. This approach allows to prove the security of a scheme under an assumption that given primitives have certain properties or – generally – assuming that certain facts are true. An example result would be to show that a given mode of operation of a block cipher is secure (i.e. that it satisfies the adopted notion of security under the adopted adversary model) assuming that the underlying cipher satisfies certain criteria (e.g. that it is a pseudo-random permutation).

The only known encryption scheme which enjoys unconditionally provable security is the One Time Pad, which does not limit its security to the key-space but to the message-space. However a significant drawback in its practical applications is the requirement for a secure channel for transmitting a message-unique key of the same size as that of the message.

Modern symmetric encryption algorithms are much easier to apply than the OTP but they do not offer provable security. None of the algorithms has also been proved to require a brute-force search of the key-space, although no better method of breaking a number of them is publicly known.

In this paper we discuss the properties of the key-setup phase of symmetric encryption schemes and their consequences for the security of the scheme. We find that a key-setup routine which is a one-way function of argument *key* and a message-unique parameter *nonce* and which satisfies the IND-CNA property (indistinguishability under chosen nonce attack) can ensure conditionally provable security against key-recovery attacks.

2 Inverting a composition of functions

Let f and g be functions: $f\colon X \to Y$ and $g\colon Y \to Z$, where g is not the inverse of f. Let $x \in X$, $y \in Y$, $z \in Z$.

Assume an adversary whose task is to find the value of x. Assume the adversary knows the value of $z = g(f(x))$ and that she can easily find the value of y given the value of $z = g(y)$.

In the described scenario the adversary is unable to accomplish her task unless she can recover x from $y = f(x)$.

In other words:

Lemma 1. *Even if an adversary can invert $g(y)$, the necessary condition for inverting $g(f(x))$ is inverting $f(x)$.*

2.1 Implications for symmetric encryption schemes

Let $Ct = E(Pt, K)$ denote a symmetric encryption scheme, where Ct and Pt represent ciphertext and plaintext data respectively, K denotes a secret key and E an encryption algorithm. Security of such scheme can be formalised in very strict terms, like IND-CPA, which gives the adversary a significantly easier task to accomplish than in the more classical, but also more demanding model – where she needs to recover the secret key K. We might informally agree that recovering K is a greater threat to the secrecy of the encrypted data than distinguishing between two chosen plaintexts by observing their ciphertexts. This is not to state that schemes only failing to satisfy IND-CPA are secure, but that those with feasible key-recovery attacks induce a more direct threat to the secrecy of Pt.

Using an analogy to Lemma 1 we can modify the model into $Ct = E(Pt, f(K))$, where f is a one-way function with a codomain compatible with the input to E. Any attack recovering the key $f(K)$ would be no threat to the secrecy of K if f was one-way. The adversary – apart from recovering the encryption key $f(K)$ – would still need to invert f to find the secret key K.

This however is obviously redundant – the knowledge of $f(K)$ is sufficient for the adversary to decrypt new messages, as she can input $f(K)$ to the decryption oracle and the value of K becomes irrelevant.

However a simple modification to the scheme would actually force the adversary to invert f before she could decrypt a new message. Assume the scheme is defined as $Ct = E(Pt, g(K, N))$, where N is a nonce (N has a different non-secret value for each encrypted message) and g is a function with a codomain compatible with the input to E such that: (a) recovering K from $g(K, N)$ is infeasible and (b) g is IND-CNA (indistinguishable under chosen nonce attack), i.e. an adversary who chooses N_1 and N_2 and observes $G_1 = g(K, N_{1+R})$ and $G_2 = g(K, N_{2-R})$, where $Prob(R = 0) = Prob(R = 1) = 1/2$, is unable to tell whether $G_1 = g(K, N_1)$ or $G_1 = g(K, N_2)$ with probability higher than 1/2.

After the modification an adversary who can perform a key-recovery attack on $Ct_1 = E(Pt_1, g(K, N_1))$ and find the value of $g(K, N_1)$, has no advantage in decrypting a new message $Ct_2 = E(Pt_2, g(K, N_2))$ unless she can invert g and find the value of K. The value of $g(K, N_1)$ is irrelevant to the value of $g(K, N_2)$ as long as the key-setup is IND-CNA.

In other words:

Lemma 2. *Let A denote a set of all attacks against the scheme $Ct_x = E(Pt_x, g(K, N_x))$, including an attack recovering the encryption key $g(K, N_x)$. Even if an adversary can perform a successful A-attack on $Ct_1 = E(Pt_1, g(K, N_1))$ and recover $g(K, N_1)$, the necessary condition for decrypting a new message $Ct_2 = E(Pt_2, g(K, N_2))$ is either performing another A-attack on Ct_2 or inverting $g(K, N_1)$ and recovering the value of K.*

Lemma 2 says that a one-way and IND-CNA key-setup delivers an additional key-setup-based level of resistance against key-recovery attacks. Recovering the secret key K requires (1) an actual attack recovering the encryption key and (2) the inverse of the key-setup routine $g(K, N)$. Without the inverse of g a successful attack recovering the encryption key $g(K, N_1)$ on one message gives the adversary no advantage in decrypting any other message encrypted with K.

3 General description of the VMPC Cryptosystem

The VMPC Cryptosystem specifies a method of transmitting j messages encrypted with the same secret key K and with a message-unique nonce N_i between two parties. The scheme is based on the VMPC Stream Cipher and its Key Scheduling Algorithm, which were presented at FSE'04 in [8]. The original KSA was modified here to be a one-way IND-CNA function.

4 VMPC Cryptosystem

Notation:

$Plaintext_i[m]$: m-th byte of i-th message
$Ciphertext_i[m]$: m-th byte of encrypted i-th message

P_i : 256-byte table storing a permutation
s : 8-bit variable
K : c-byte table storing the secret key; $16 \leq c \leq 64$
N_i : z-byte table storing the nonce of i-th message; $16 \leq z \leq 64$
$+$: addition modulo 256

1. Parties A and B set up the key K, which is known only to A and B
2. Party A transmits j messages to Party B by following steps 3-8:

3. For i from 1 to j execute steps 4-8:

4. Generate a nonce N_i such that $N_i \neq N_d$ for any $d \neq i$
5. Input K and N_i to VMPC KSA by executing steps 5.1 - 5.4.2
   $[P_i = KSA(KSA(KSA(P^{(0)}, K), N_i), K)]$:

   5.1. $s = 0$; for n from 0 to 255: $P^{(0)}[n] = n$; $P_i = P^{(0)}$
   5.2. for m from 0 to 767: execute steps 5.2.1 - 5.2.2:
      5.2.1. $s = P_i[s + P_i[m \bmod 256] + K[m \bmod c]]$
      5.2.2. $x = P_i[m \bmod 256]$; $P_i[m \bmod 256] = P_i[s]$; $P_i[s] = x$
   5.3. for m from 0 to 767: execute steps 5.3.1 - 5.3.2:
      5.3.1. $s = P_i[s + P_i[m \bmod 256] + N_i[m \bmod z]]$
      5.3.2. $x = P_i[m \bmod 256]$; $P_i[m \bmod 256] = P_i[s]$; $P_i[s] = x$
   5.4. for m from 0 to 767: execute steps 5.4.1 - 5.4.2:
      5.4.1. $s = P_i[s + P_i[m \bmod 256] + K[m \bmod c]]$
      5.4.2. $x = P_i[m \bmod 256]$; $P_i[m \bmod 256] = P_i[s]$; $P_i[s] = x$

6. Encrypt i-th message with VMPC Stream Cipher by executing steps 6.1-6.1.3:
   $[Ciphertext_i = Plaintext_i \text{ xor } SC(P_i)]$

   6.1. For m from 0 to ($Length$ of i-th message)-1 execute steps 6.1.1 - 6.1.3:
      6.1.1. $s = P_i[s + P_i[m \bmod 256]]$
      6.1.2. $Ciphertext_i[m] = Plaintext_i[m] \text{ xor } P_i[P_i[P_i[s]] + 1]$
      6.1.3. $x = P_i[m \bmod 256]$; $P_i[m \bmod 256] = P_i[s]$; $P_i[s] = x$

7. Prepend N_i to $Ciphertext_i$
8. Transmit $Ciphertext_i$ to Party B

5 Conditional proof of security of VMPC Cryptosystem

Note: determining X is used here to denote determining any information about the value of X with probability higher than expected from a random guess.

Definition 1. *Breaking the cryptosystem: Determining* $Plaintext_i$ *given* $Ciphertext_i$ *and given* $Ciphertext_d$ *and* $Plaintext_d$ *for any values of* $d \neq i$

Definition 2. *Attack on the cipher: An algorithm recovering the internal state* P_d *from the keystream* $SC(P_d)$*, where* $SC(P_d) = Ciphertext_d$ *xor* $Plaintext_d$

Observation 1. According to theoretical analyses and statistical tests described in [8], there is no known method of distinguishing the output of the VMPC Stream Cipher ($SC(P_i)$) from a truly random data stream.

Observation 2. According to [8], one phase of the KSA (768 iterations in steps 5.2 or 5.3 or 5.4) provides an undistinguishable from random diffusion [1] of changes of one bit or byte of key of size up to 64 bytes onto the generated permutation P_i and onto output generated by the VMPC Cipher ($SC(P_i)$ in step 6.1). The double repetition of the 768 steps of the KSA in steps 5.3 and 5.4 provides the diffusion effect with an extensive safety margin. From these results we conjecture that the transformation $P_i = KSA(KSA(KSA(P^{(0)}, K), N_i), K)$, as specified in steps 5.1-5.4, provides the IND-CNA property, defined in Section 2.1.

Assumption 1. Let Adv_1 denote the probability that an adversary who has no information about the value of $SC(P_i)$ finds the value of $Plaintext_i$, where $Plaintext_i = Ciphertext_i$ xor $SC(P_i)$.

$$Adv_1 = 2^{-8 \times c+1}$$

where c is the size of the secret key K in bytes.

Elucidation. According to Observation 1, the value of $SC(P_i)$ cannot be distinguished from a truly random value. As a result the value of $Ciphertext_i = Plaintext_i$ xor $SC(P_i)$ is also undistinguishable from random, which thwarts the possibility of deducing any information about $Plaintext_i$ directly from $Ciphertext_i$.

Out of the remaining sources of information in the described cryptosystem $SC(P_i)$ appears to be the only value sufficiently related to $Plaintext_i$ to enable determining $Plaintext_i$.

[1] Relations between $P_1 = KSA(P^{(0)}, K_1)$ and $P_2 = KSA(P^{(0)}, K_2)$ and relations between $SC(P_1)$ and $SC(P_2)$, where K_1 differs from K_2 in one bit or one byte, are undistinguishable from relations expected between – respectively – two random permutations or two random data-streams

Assumption 2. Let Adv_2 denote the probability that an adversary who has no information about the value of P_i finds the value of $SC(P_i)$, where $SC(P_i)$ is the cipher's output derived in step 6.1.

$$Adv_2 = 2^{-8\times c+1}$$

Elucidation. According to Observation 1, the value of $SC(P_i)$ cannot be distinguished from a truly random value. As a result the value of $Ciphertext_i = Plaintext_i$ xor $SC(P_i)$ is also undistinguishable from random, which thwarts the possibility of deducing any information about $SC(P_i)$ directly from $Ciphertext_i$.

According to Observation 2, relations between the values of $SC(P_i)$ and $SC(P_d)$ and between the values of P_i and P_d are undistinguishable from random, which thwarts the possibility of deducing any information about $SC(P_i)$ from $SC(P_d)$ for $d \neq i$.

Out of the remaining sources of information in the described cryptosystem P_i appears to be the only value sufficiently related to $SC(P_i)$ to enable determining $SC(P_i)$.

Assumption 3. Let Adv_3 denote the probability that an adversary who has no information about the value of K finds the value of P_i, where
$P_i = KSA(KSA(KSA(P^{(0)}, K), N_i), K)$.

$$Adv_3 = 2^{-8\times c+1}$$

Elucidation. According to Observation 2, relations between the values of $P_i^{(2)} = KSA(KSA(P^{(0)}, K), N_i)$ and $P_d^{(2)} = KSA(KSA(P^{(0)}, K), N_d)$ and, as a result of step 5.4, relations between the values of P_i and P_d are undistinguishable from random for $N_i \neq N_d$, which thwarts the possibility of determining P_i from P_d for $d \neq i$.

Out of the remaining sources of information in the described cryptosystem K appears to be the only value sufficiently related to P_i to enable determining P_i.

Assumption 4. Let Adv_4 denote the probability that an adversary who has no information about the value of P_d finds the value of K, where
$P_d = KSA(KSA(KSA(P^{(0)}, K), N_d), K)$ for any value of $d \in \{1, 2, \ldots, j\}$.

$$Adv_4 = 2^{-8\times c+1}$$

Elucidation. In the described cryptosystem K is used nowhere else than for the computation of the value of $P_d = KSA(KSA(KSA(P^{(0)}, K), N_d), K)$ for $d \in \{1, 2, \ldots, j\}$.

Lemma 3. *Even if an adversary performs a successful attack on the cipher and recovers P_d from $SC(P_d)$, to break the cryptosystem and determine $Plaintext_i$ for $i \neq d$, she needs to invert $P_d = KSA(KSA(KSA(P^{(0)}, K), N_d), K)$ and recover the value of K, where N_d is a known parameter.*

Lemma 3 is true provided that assumptions 1-4 are true. Following Assumption 1, breaking the cryptosystem cannot be accomplished without determining $SC(P_i)$; following Assumption 2, $SC(P_i)$ cannot be determined without determining P_i; following Assumption 3, P_i cannot be determined without determining K; following Assumption 4, K can only be recovered from P_d for $d \in \{1, 2, \ldots, j\}$. If P_d is known, obviously for $d \neq i$, which is assumed to be true as a result of a successful attack, then recovering [2] K from P_d is a necessary condition for breaking the cryptosystem. □

5.1 The problem of recovering K from P_d

The transformation $P_d = KSA(KSA(KSA(P^{(0)}, K), N_d), K)$ performs 2304 operations, each constituting (5.X.1) an update of the s variable through combining a consecutive byte of K, the previous value of s and two elements of P_d and (5.X.2) a swap operation of a consecutive-modulo-256 and s-th element of P_d. As a result each element of P_d undergoes an average number of 18 K-dependant changes in the course of the transformation. We believe that tracking these operations in the purpose of providing a proof of their uninvertibility is practically unreachable.

On the other hand the extent of the mixing the transformation performs on P_d indicates that the transformation might actually be infeasible to invert.

If a 2-phase KSA $P'_d = KSA(KSA(P^{(0)}, K), N_d)$ was used, inverting it could be reduced to the problem of recovering $KSA(P^{(0)}, K)$. Recovering K would be redundant because the known values $KSA(P^{(0)}, K)$ and N_i would allow to reconstruct $P'_i = KSA(KSA(P^{(0)}, K), N_i)$, generate the keystream $SC(P'_i)$ and recover $Plaintext'_i = Ciphertext'_i$ xor $SC(P'_i)$. This would be possible since given $P'_d = KSA(KSA(P^{(0)}, K), N_d)$ and the known value of N_d the operations performed by the KSA could be backtracked to recover $KSA(P^{(0)}, K)$ with the guess-work factor limited only to the value of s.

However as a result of the third phase of the KSA (step 5.4), the guess-work required by the backtracking algorithm would be significantly higher than that because both the value of K and the value of $P_d^{(2)} = KSA(KSA(P^{(0)}, K), N_d)$ in $P_d = KSA(P_d^{(2)}, K)$ would be unknown. Each iteration of the third phase of the KSA usually uses one byte of K and 3 elements of the $P_d^{(2)}$ permutation, all of which are unknown, which implies that usually 4 guesses per each iteration

[2] recovering rather than determining here – the knowledge of all the bits of K is necessary in consequence of the KSA diffusion effect described in Observation 2

are required to proceed with the backtracking. This would result in more than $(c+1)$ guessed values after c iterations (c denoting the length of K in bytes), which is not more efficient than searching through all the $2^{8\times c}$ possible values of K.

From the above observations we conjecture Assumption 5:

Assumption 5. The most efficient method of recovering K from $P_d = KSA(P_d^{(2)}, K) = KSA(KSA(KSA(P^{(0)}, K), N_d), K)$ is searching through the $2^{8\times c}$ possible values of K until one of them conforms to $P_d = KSA(KSA(KSA(P^{(0)}, K), N_d), K)$, where P_d is assumed to be known as a result of a successful attack and N_d is a known parameter.

Provided that Assumption 5 is true, we can formulate a lemma:

Lemma 4. *Searching through the possible $2^{8\times c}$ values of K is a necessary condition for breaking the cryptosystem.*

Lemma 4 is a direct consequence of Lemma 3, saying that recovering K from $P_d = KSA(KSA(KSA(P^{(0)}, K), N_d), K)$ is a necessary condition for breaking the cryptosystem, and Assumption 5, which says that recovering K from P_d requires a brute-force search for the correct value of K. $\square$

6 Consequences of the proof for the cryptosystem

The key component of the cryptosystem is the third phase of the Key Scheduling Algorithm (step 5.4) without which the cryptosystem has a standard level of security, where determining a new plaintext is easily possible if a practical algorithm recovering the internal state from the keystream is found.

In the proposed cryptosystem it is assumed that the adversary can recover the cipher's internal state P_d for one message, but still – because of the third K-dependant phase of the KSA and under the specified assumptions – the scheme can be proved secure since the internal state P_i used to encrypt any other message with the same key is pseudorandomly different from P_d, following the IND-CNA property, and the KSA transformation is a one-way function.

In consequence the cryptosystem offers what might be viewed as a two-layer level of security, where the additional layer is derived strictly from the properties of the one-way IND-CNA key-setup: recovering the internal state P_d from the cipher's output $SC(P_d)$ is still believed to require an average computational effort of about 2^{900} operations, as estimated in [8], but even if this is overcome, breaking the cryptosystem would additionally require a feasible algorithm recovering the secret key K from the internal state $P_d = KSA(KSA(KSA(P^{(0)}, K), N_d), K)$.

7 VMPC Cryptosystem test vectors

Table 1 gives 16 test output-bytes generated by the VMPC Cryptosystem for a given 16-byte key K and a given 16-byte nonce N:

Table 1. Test-outputs of the VMPC Cryptosystem

K [hex]	96, 61, 41, 0A, B7, 97, D8, A9, EB, 76, 7C, 21, 17, 2D, F6, C7
N [hex]	4B, 5C, 2F, 00, 3E, 67, F3, 95, 57, A8, D2, 6F, 3D, A2, B1, 55

Output-byte position [dec]	0	1	2	3	252	253	254	255
Output-byte value [hex]	B6	EB	AE	FE	48	17	24	73
Output-byte position [dec]	1020	1021	1022	1023	102396	102397	102398	102399
Output-byte value [hex]	1D	AE	C3	5A	1D	A7	E1	DC

8 Conclusions

We analysed the role of the properties of the key-setup phase in symmetric encryption schemes and their consequences for the security of the scheme against attacks recovering the encryption key. We found that designing one-way key-setup procedures satisfying the IND-CNA property for nonce-based cryptosystems can provide a noticeable improvement in security against key-recovery attacks.

We presented a proof of security of a specific cryptosystem based on both the properties of the one-way IND-CNA key-setup and on a set of scheme-specific assumptions. Although the assumptions relate to many components of the cryptosystem they were intended to be relatively acceptable in practice as not to significantly diminish at least the practical value of the conditional proof.

Most of all the paper was intended to illustrate that a nonce-based key-setup routine meeting certain criteria allows to build symmetric encryption schemes with an additional key-setup-based layer of resistance against the general class of key-recovery attacks.

References

1. Mihir Bellare: Practice-Oriented Provable Security, Proceedings of First International Workshop on Information Security, LNCS vol. 1396, Springer-Verlag 1999.
2. Jaechul Sung, Sangjin Lee, Jongin Lim, Seokhie Hong, Sangjoon Park: Provable Security for the Skipjack-like Structure against Differential Cryptanalysis and Linear Cryptanalysis, Proc. of ASIACRYPT 2000, LNCS vol. 1976, Springer-Verlag 2000.

3. Ju-Sung Kang, Sang-Uk Shin, Dowon Hong, Okyeon Yi: Provable Security of KASUMI and 3GPP Encryption Mode f8, Proceedings of ASIACRYPT 2001, LNCS vol. 2248, Springer-Verlag 2001.
4. Ross Anderson, Eli Biham: The practical and Provably Secure Block Ciphers: BEAR and LION, Proceedings of Fast Software Encryption 1996, LNCS vol. 1039, Springer-Verlag 1996.
5. Mihir Bellare, Anand Desai, Eron Jokipii, Phillip Rogaway: A Concrete Security Treatment of Symmetric Encryption, Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS '97), IEEE 1997.
6. Mihir Bellare, Philip Rogaway, David Wagner: The EAX Mode of Operation Proceedings of Fast Software Encryption 2004, LNCS vol 3017, Springer-Verlag 2004.
7. Tadayoshi Kohno, John Viega, Doug Whiting: CWC: A High-Performance Conventional Authenticated Encryption Mode, Proceedings of Fast Software Encryption 2004, LNCS vol 3017, Springer-Verlag 2004.
8. Bartosz Zoltak: VMPC One-Way Function and Stream Cipher, Proceedings of Fast Software Encryption 2004, LNCS vol 3017, Springer-Verlag 2004.