A Chosen Plaintext Attack on Offset Public Permutation Mode

Miloslav Homer
miloslav.homer@gmail.com

April 15, 2018

Abstract

Offset Public Permutation Mode (OPP) by Granger et al. is a one-pass authenticated encryption scheme supporting associated data (AEAD scheme). Leveraging an error in analysis of the scheme, a chosen plaintext attack that creates a forgery was discovered. This attack makes no assumptions about the underlying tweakable blockcipher while having negligible complexity requirements and high probability of success. An implementation of the attack is also provided.
Keywords: symmetric cryptography, block cipher mode of operation, authenticated encryption, forgery attack, distinguishing attack.

1 Introduction

Authenticated Encryption (AE) refers to providing both confidentiality and authenticity. These schemes are symmetric, meaning that a single key is used for both encryption and decryption. Often it is required that such scheme supports Associated Data (also called Header) – data whose authenticity is ensured but which are not encrypted (such schemes are AEAD schemes). These schemes have huge practical applications, providing a simple interface to accomplish two security goals.

In [1] a new single pass AEAD scheme named Offset Public Permutation Mode (OPP) was published. This scheme offers high speed of encryption (peaking at 0.55 cycles per byte) and full paralelization. If such scheme also had a valid proof of security, it would be very attractive for all applications.

Unfortunately, the authors of OPP made an error in their analysis. This error lead to a chosen plaintext attack with negligible complexity requirements and high probability of success. This attack is presented in this paper. Originally, this attack was discovered in [2] (in Slovak). This paper completes and expands it.

2 Notation

Denote by $\mathbb{F}_{2^n}$ the finite field of order 2^n with $n \geq 1$. A b-bit string X is an element of $\{0,1\}^b$ (or equivalently of the $\mathbb{F}_2$-vector space $\mathbb{F}_2^b$). The length of such string X is denoted by $|X|$ (for $X \in \{0,1\}^b$ holds $|X| = b$). The bit string of length 0 is denoted by ϵ. The concatenation of two bit strings X, Y is denoted by $X\|Y$. Symbol $\oplus$ denotes bit-wise XOR.

Given a b-bit string $X = x_0\| \ldots \|x_{b-1}$ we define $\text{first}_l(X) = x_0\| \ldots \|x_{l-1}$ (for $1 \leq l \leq b$). Denote $\text{pad}_b^A(Y)$ the padding function:

$$\text{pad}_b^A(Y) = Y\|A\|0^{b-|Y|-|A|}.$$

The parameters $b, k, n, \tau \geq 0$ denote block size, key lenght, nonce length and tag length, respectively. In OPP it is required that $n \leq b - k - 1$.

3 Preliminaries

3.1 Tweakable Blockciphers

Let $\mathcal{T}$ be a set of "tweaks". A tweakable blockcipher $E\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^b \to \{0,1\}^b$ is a function such that for every $K \in \{0,1\}^k$ and every tweak $T \in \mathcal{T}$ function $E(K,T,.)\colon \{0,1\}^b \to \{0,1\}^b$ is a permutation. Inverse permutation is denoted $E^{-1}(K,T,.)$.

A tweakable permutation $\pi\colon \mathcal{T}\{0,1\}^b \to \{0,1\}^b$ is a function such that for every $T \in \mathcal{T}$ function $\pi(T,.)\colon \{0,1\}^b \to \{0,1\}^b$ is a permutation.

Granger et al. [1] proposed Mixed tweakable pseudorandom permutation (MTPRP) security as a security definition for a tweakable blockcipher. It is a middle ground between tweakable pseudorandom permutation (TPRP) notion and strong TPRP (STPRP) notion.

- In the former, the adversary is permitted encryption queries only.
- In the latter, the adversary is also permitted decryption queries.
- In MTPRP, we utilize a partition of tweakspace, say $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1$. For tweaks in $\mathcal{T}_0$ the adversary can only ask encryption queries while for tweaks in $\mathcal{T}_1$ the adversary can also ask decryption queries.

The goal of the adversary is to distinguish between a tweakable blockcipher with randomly chosen key and a randomly chosen tweakable permutation. We say that E is a secure MTPRP if the advantage:

$$\mathrm{Adv}^{MTPRP}(A) = \left|\Pr\left[A^E = 1\right] - \Pr\left[A^\pi = 1\right]\right|$$

is negligible for every bounded adversary A, where $A^E = 1$ denotes that the adversary interacted with a tweakable blockcipher with randomly chosen key and the adversary returned it interacted with a tweakable blockcipher with an unknown key. The symbol $A^\pi = 1$ denotes that the adversary interacted with a randomly chosen tweakable permutation and the adversary returned it interacted with a tweakable blockcipher with an unknown key.

3.2 Nonce-based Authenticated Encryption (with Associated Data)

Denote the associated data H and the key K. OPP employs a nonce – this value should not be repeated for a given key, although an adversary can know and predict it (denoted N). A tag is a short string appended to the ciphertext (denoted T). Formally, an abstract Authenticated Encryption with Associated Data (AEAD) scheme utilizing nonces operates as follows:

$$\mathcal{E}_K(N,H,M) = (C,T),$$
$$\mathcal{D}_K(N,H,C,T) = M/\perp.$$

If for given (N,H,C,T) there doesn't exist a message M such that $\mathcal{E}_K(N,H,M) = (C,T)$ then the decryption algorithm outputs $\perp$ indicating that such ciphertext C with tag T is invalid given nonce N, associated data H and key K.

Following [3]: an ideal counterpart to this scheme is a random oracle that for a plaintext of length m, a nonce and associated data returns a string of random bits of length $m + \tau$, where τ is the lenght of the tag.

Distinguishing between such an oracle and a real scheme with random key is the goal of the adversary – denoted PRIV as in [3]. The advantage of such adversary is defined as:

$$\mathrm{Adv}^{PRIV}_{\mathrm{AEAD}}(A) = \left|\Pr\left[A^{\mathrm{AEAD}} = 1\right] - \Pr\left[A^{\$} = 1\right]\right|.$$

In PRIV there are additional restrictions placed on the adversary: the adversary can only use encryption queries and cannot repeat nonces.

Granger et al. [1] relaxed these restrictions for the OPP scheme. The adversary can use decryption queries – a random oracle would always return $\perp$ (i.e. the ciphertext with tag are always invalid). The adversary is nonce-respecting in the following sense, citing [1], Appendix B: "Let D be a *nonce-respecting* AE distinguisher against OPP, which means that it never makes an encryption query for a nonce that was used before." Therefore, in their setting it is possible to ask multiple decryption queries with a previously used nonce, provided that only one encryption query for a given nonce was asked. One natural restriction was added: the adversary cannot query a text for decryption if such text was recieved as an output of an encryption query. The advantage of an adversary is defined as:

$$\mathrm{Adv}^{Granger}_{\mathrm{AEAD}}(A) = \left|\Pr\left[A^{\mathrm{AEAD}} = 1\right] - \Pr\left[A^{\$} = 1\right]\right|.$$

Granger et al. do not claim anything else about OPP. In contrast, Rogaway defines a second experiment, denoted AUTH. In AUTH, the attacker interacts with real scheme with random key. The attacker is tasked to produce a forgery – value (N, H, C, T). If $D_K(N, H, C, T) \neq \perp$, the attacker is successful. The advantage of such adversary is defined as:

$$\mathrm{Adv}^{AUTH}_{\mathrm{AEAD}}(A) = \Pr\left[D_K(A^{\mathrm{AEAD}}) \neq \perp\right].$$

The adversary is restricted:

- The adversary can only ask encryption queries.
- The forgery attempt cannot be an output of an encryption query.
- The adversary is nonce-respecting: two encryption queries cannot have identical nonce.

Note that the adversary can reuse a nonce in his forgery attempt.

4 Offset Public Permutation Mode

OPP mode defined in [1] utilizes tweakable blockciphers. Since the attack described in this article does not assume anything about the tweakable blockcipher, we do not need to concern ourselves with the details of tweakable blockcipher construction. Instead, it is assumed that a MTPRP secure abstract tweakable blockcipher is used with tweakspace $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1$ such that $(X, 0, 0, 0) \in \mathcal{T}_0$ for all $X \in \{0,1\}^b$ (see subsection 3.1, and assumptions of theorem 2 in [1]). The condition of $(X, 0, 0, 0) \in \mathcal{T}_0$ for all $X \in \{0,1\}^b$ is satisfied: since such tweaks are used only in one line (see line 4 in function `OPPAbs` in figure 1) – such tweaks are not used in inverse direction. There are no assumptions about $\mathcal{T}_1$.

The tweaks are of the form $\{0,1\}^{b-k} \times \mathbb{N}^3$. Calling a tweakable blockcipher encryption on plaintext P with key K and tweak (X, i_0, i_1, i_2) is denoted by $E^{i_0,i_1,i_2}_{K,X}(P)$. In similar fashion a tweakable blockcipher decryption is denoted by $D^{i_0,i_1,i_2}_{K,X}(C)$

OPP mode is defined via algorithms `OPPEnc`, `OPPDec` and `OPPAbs`. Algorithms `OPPEnc` and `OPPAbs` are used in the encryption routine `OPPE`, algorithms `OPPDec` and `OPPAbs` are then used in the deryption routine `OPPD`.

Function `OPPEnc`(K, X, M)**:**

$M_0 \| M_1 \| \ldots \| M_{m-1} \leftarrow M$, st. $|M_i| = b, 0 \leq |M_{m-1}| < b$;
$C \leftarrow \epsilon$;
$S \leftarrow 0^b$;
for $i \in \{0, 1, \ldots m-2\}$ **do**
  $C_i \leftarrow E^{i,0,1}_{K,X}(M_i)$;
  $C \leftarrow C \| C_i$;
  $S \leftarrow S \oplus M_i$;
if $|M_{m-1}| > 0$ **then**
  $Z \leftarrow E^{m-1,1,1}_{K,X}(0^b)$;
  $C_{m-1} \leftarrow \text{first}_{|M_{m-1}|}\left(\text{pad}^0_b(M_{m-1}) \oplus Z\right)$;
  $C \leftarrow C \| C_{m-1}$;
  $S \leftarrow S \oplus \text{pad}^{10}_b(M_{m-1})$;
return C, S

Function `OPPAbs`(K, X, H, S, l)**:**

$H_0 \| H_1 \| \ldots \| H_{h-1} \leftarrow H$, st. $|H_i| = b, 0 \leq |H_{h-1}| < b$;
$S' \leftarrow 0^b$;
for $i \in \{0, 1, \ldots h-2\}$ **do**
  $S' \leftarrow S' \oplus E^{i,0,0}_{K,X}(H_i)$;
if $|H_{h-1}| > 0$ **then**
  $S' \leftarrow S' \oplus E^{h-1,1,0}_{K,X}(\text{pad}^{10}_b(H_{h-1}))$;
$j \leftarrow \lceil l/b \rceil + 2$;
return $\text{first}_t\left(S' \oplus E^{h-1,j,0}_{K,X}(S)\right)$

Function `OPPDec`(K, X, C)**:**

$C_0 \| C_1 \| \ldots \| C_{m-1} \leftarrow C$, st. $|C_i| = b, 0 \leq |C_{m-1}| < b$;
$M \leftarrow \epsilon$;
$S \leftarrow 0^b$;
for $i \in \{0, 1, \ldots m-2\}$ **do**
  $M_i \leftarrow D^{i,0,1}_{K,X}(C_i)$;
  $M \leftarrow M \| M_i$;
  $S \leftarrow S \oplus M_i$;
if $|C_{m-1}| > 0$ **then**
  $Z \leftarrow E^{m-1,1,1}_{K,X}(0^b)$;
  $M_{m-1} \leftarrow \text{first}_{|C_{m-1}|}\left(\text{pad}^0_b(C_{m-1}) \oplus Z\right)$;
  $M \leftarrow M \| M_{m-1}$;
  $S \leftarrow S \oplus \text{pad}^{10}_b(M_{m-1})$;
return M, S

Function `OPPE`(K, N, H, M)**:**

$X \leftarrow \text{pad}^0_{b-n-k}(N)$;
$C, S \leftarrow$`OPPEnc`(K, X, M);
$T \leftarrow$`OPPAbs`$(K, X, H, S, |M| \mod b)$;
return C, T;

Function `OPPD`(K, N, H, C, T)**:**

$X \leftarrow \text{pad}^0_{b-n-k}(N)$;
$M, S \leftarrow$`OPPDec`(K, X, C);
$T' \leftarrow$`OPPAbs`$(K, X, H, S, |M| \mod b)$;
if $T = T'$ **then**
  return M
else
  return $\perp$

Figure 1: Offset Public Permutation Mode (OPP)

5 The Attack

5.1 Description

The attack is described as a routine of an adversary – the adversary performs the steps. Additionally, the adversary can query the encryption/decryption oracle. To avoid complicated formalism this action is denoted as a function call. The task of the adversary is to distinguish between a real oracle and a random oracle (see subsect. 3.2).

Algorithm 1: Routine creating a forgery using one chosen plaintext request.

Function `CreateForgery`$(\mathcal{E}, N, H, p)$**:**

1. $M_1 \leftarrow 0^{2b+p}$;
2. $C, T \leftarrow \mathcal{E}(N, H, M_1)$;
3. $C_1 \| C_2 \| C_3 \leftarrow C$, st. $|C_1| = b, |C_2| = b, |C_3| = p$;
4. **return** $N, H, C_1 \| 0^p, T$

A routine (see algorithm 1) that produces forgeries with significant probability of validity was discovered in [2]. A forgery consists of a nonce, a header and a ciphertext with tag. This routine can be used for every valid header and valid nonce, so these are specified as the parameters of the routine. The last parameter, $1 \leq p \leq t$, specifies the length of the last block used in the attack. To achieve optimal theoretical probability of generating a valid forgery set $p = 1$.

Proposition 1. *Let $\mathcal{E}$ be the oracle providing encryption routine of OPP using an ideal tweakable blockcipher. Let N be a valid OPP nonce and let H be a valid OPP header. Let $1 \leq p \leq t$. Then the routine in algorithm 1 called with parameters $\mathcal{E}, N, H, p$ returns a valid forgery with probability 2^{-p}.*

Proof. Denote the output of the query $O = C_1 \| C_2 \| C_3, T$, where $|C_1| = |C_2| = b$ and $|C_3| = p$.

This proof has three parts:

1. Showing that C_1 decrypts to 0^b.
2. Verifying that 0^p (last block) decrypts to 0^p with probability 2^{-p}.
3. Given the last block (0^p) decrypts to 0^p, showing that the result of decryption is not $\perp$.

The first step is trivial, since the nonce and the key used are identical. Second step: 0^p is decrypted by XOR-ing with p random bits (see line 10 in `OPPDec` in figure 1). The bits are random since it is assumed that the tweakable blockcipher used is ideal and the tweak $(X, 0, 1, 1)$ was not used when computing the query.

To show the third step, we only need to show that value S computed in `OPPEnc`/`OPPDec` and used in `OPPAbs` is equal "in query" and "in forgery verification". In query:

$$S = M_1 \oplus M_2 \oplus (0)^p 10(0)^{b-p-2} = 0^b \oplus 0^b \oplus (0)^p 10(0)^{b-p-2} = (0)^p 10(0)^{b-p-2}.$$

Given that 0^p is decrypted to 0^p in forgery verification:

$$S = 0^b \oplus (0)^p 10(0)^{b-p-2} = (0)^p 10(0)^{b-p-2}.$$

Since parameters in `OPPAbs` are identical "in query" and "in forgery verification", the output of `OPPAbs` when decrypting is equal to T. Therefore $\perp$ is not returned. $\square$

Therefore routine `CreateForgery` can be used to create a valid forgery with significant probability. Therefore OPP cannot satisfy the AUTH requirement for nonce-based AEAD schemes. However, Granger et al. do not claim that OPP satisfies this requirement.

This routine can also be used to distinguish between a random and a real oracle. To avoid unnecessary formalism, it is assumed that the adversary is provided access to two oracles – an encryption oracle and a decryption oracle.

Algorithm 2: Distinguishing between a real and a random oracle via forgery.

Function `Distinguish`$(\mathcal{E}, \mathcal{D}, N, H, p)$**:**

1 $(N, H, F, T) \leftarrow$`CreateForgery`$(\mathcal{E}, N, H, p)$;
2 $o \leftarrow \mathcal{D}(N, H, F, T)$;
3 **if** $o = \perp$ **then**
4 **return** *0*
5 **else**
6 **return** *1*

In algorithm 2 we can see an example of an attacker using the ability to create forgery to distinguish between a real and a random oracle. Again, the nonce and the header can be chosen arbitrarily, so they are a parameter of the distinguisher.

Proposition 2. *Let $\mathcal{E}, \mathcal{D}$ be either the oracle providing encryption/decryption routine of OPP using an ideal tweakable blockcipher or corresponding random oracles. Let N be a valid OPP nonce and let H be a valid OPP header. Let $1 \leq p \leq t$. Then the routine in algorithm 2 called with parameters $\mathcal{E}, \mathcal{D}, N, H, p$ has advantage greater or equal to 2^{-p}.*

Proof. To determine the advantage, we have to determine two values: the probability that the distinguisher returns 1 when interacting with the real oracle and the probability that the distinguisher returns 1 when interacting with the random oracle.

Starting with the latter, this probability is equal to zero. The distinguisher only returns 1 when the decryption oracle returns a plaintext and not $\perp$. Since the decryption random oracle always returns $\perp$, this case is clear.

Now for the former: given that $\mathcal{E}$ is a real oracle we get from proposition 1 that routine `CreateForgery` produces a valid forgery with probability 2^{-p}. This is the probability that this adversary correctly identifies a real oracle.

To conclude the proof recall the definition of the advantage:

$$\mathrm{Adv}_{\mathrm{AEAD}}^{Granger}(A) = \left|\Pr\left[A^{\mathrm{AEAD}} = 1\right] - \Pr\left[A^{\$} = 1\right]\right| = |2^{-p} - 0| = 2^{-p}.$$

□

By choosing $p = 1$ the advantage would be lower bounded by 0.5. Hence there exists an adversary with significant advantage. This result is in direct contradiction with claims of Granger et al.

Moreover, if the attacker was permitted to check all options for the last block of the ciphertext, it could produce a valid forgery with probability equal to one. This attack would need only one encryption query and 2^p decryption queries – this should be possible within the restrictions of the experiment (see section 3.2).

There is another, perhaps more practical variant. Choose a nonce N and a header H. Choose plaintext blocks $P_1, \ldots, P_m$ such that the last (m-th) block is short and that there exists i such that

$$\bigoplus_{j=i}^{m-1} P_j = 0.$$

Assume that the corresponding ciphertext is $C_1, \ldots, C_m, T$. Then to create a forgery with high probability of validity, omit blocks i to $m-1$ from the ciphertext.

5.2 Implementation

Authors of OPP provided an implementation of OPP in C on github [4]. A variation of this attack using their code as a target is implemented in C, source code is accessible on github [5].

Due to efficiency reasons, all variables (key, nonce, header, plaintext, ciphertext) are treated as a sequence of bytes, not bits. Therefore when using given implementation of OPP the smallest nonzero last block has the length of one byte. Therefore, given one encryption query and one decryption query, our probability of producing a valid forgery is $1/256$.

However, we are permitted to check all 256 possible options within the restrictions of the experiment. Therefore, it is possible to distinguish between a random oracle and a real oracle with probability equal to one. Such an attack utilizes only very small complexity requirements. An implementation of this variant is provided – checking all possible bytes of the last block to create a forgery.

6 Oversights in original analysis

Granger et al. noticed the similarities of OPP and Offset Code Book 3 (OCB3) by Krovetz and Rogaway [6]. In their appendix B Granger et al. claim that bounds of OCB3 directly apply to OPP.

This attack would not work on OCB3, because there is a subtle difference between OPP and OCB3. In OCB3 the tweak used when generating the tag depends on the length of the plaintext. The presented attack is therefore invalid, since the tweaks used when generating the tag are different. Therefore the tags would coincide with negligible probability, whereas in this attack the tags would coincide with probability that depends on the lenght of the last incomplete block.

Therefore the original conclusion that OPP generalizes OCB3 is invalid.

7 Conclusion

Offset Public Permutation Mode was a promising fast one-pass AEAD scheme that is fully paralelizable. In this paper a chosen plaintext attack on OPP was presented. Given any valid header and nonce it is possible to create a forgery with high probability of success utilizing only one encryption query, by ommiting blocks at the end of the plaintext that sum to zero. This also means that it is possible to distinguish OPP from a random oracle. A proof of concept implementation against OPP implementation is also provided. This attack was made possible by an incorrect assumption that OPP generalizes OCB3. The difference between the two is that the tweak used in tag generation in OPP does not depend on the ciphertext length.

References

[1] Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 263–293. Springer, 2016.

[2] Miloslav Homer. Jedno-priechodové schémy autentizovaného šifrovania, 2018.

[3] Phillip Rogaway. Authenticated-encryption with associated-data. In *Proceedings of the 9th ACM conference on Computer and communica tions security*, pages 98–107. ACM, 2002.

[4] Mem family of aead schemes (2015). `https://github.com/MEM-AEAD`.

[5] Proof of concept attack on opp scheme (2018). `https://github.com/ArcHound/AttackOnOPP`.

[6] Ted Krovetz and Phillip Rogaway. The software performance of authenticated-encryption modes. In *International Workshop on Fast Software Encryption*, pages 306–327. Springer, 2011.