SECURITY ANALYSIS OF PARALLELIZABLE INTEGRITY-AWARE ENCRYPTION TECHNIQUE

POOJITA SURI
GTID: 903291552
Georgia Institute of Technology,
Atlanta, GA
poojita.suri@gatech.edu

RIYA RUPIN SHAH
GTID:903340068
Georgia Institute of Technology,
Atlanta, GA
riya@gatech.edu

Abstract

In today's data driven world, speed is of utmost importance and thus the need for fast encryption and authentication is eminent. Parallelizable Integrity Aware Encryption Technique is a patented authenticated encryption scheme built to provide faster processing and overcome the drawbacks of some existing serial and parallel schemes. The inventor has detailed the workings of the scheme in the patent, but no formal security analysis has been provided. Our goal in this paper is to evaluate the security of the scheme. At first, we focus on security under classical models – IND-CPA and INT-CTXT. These models often make use of assumptions that do not always hold in the real world. Given this gap between theoretical assumptions made in these traditional models and real-world settings in which these schemes are deployed, we shift focus on the impact of repeating nonces and evaluate this scheme in the RUP setting.

Keywords
Authenticated encryption, Parallelizable, Integrity Aware, IND-CPA, INT-CTXT, IND-CCA, Nonce-reuse, INT-RUP

I. Introduction and Motivation

Parallelizable Integrity Aware encryption technique (referred to as PIAET in the rest of the paper) is an authenticated encryption technique invented and patented by Markus Leech in 2005. His goal was to build high speed encryption scheme that also provides authenticity and to overcome the drawbacks of other schemes such as XCBC which has an inherent serial block chain nature, and Offset Code Book (OCB) which relies on Gray code – something that makes cryptographers skeptical [1].

II. Scheme Overview

We describe the scheme in parts - first, how the mask values are generated, then how plaintext blocks are processed and finally, how the tag is generated.

NOTATIONS USED

To be consistent with the patent we evaluated, these are the notations we have used throughout the paper:

$\boldsymbol{B}$ – an entire plaintext
$\boldsymbol{B}[\boldsymbol{i}]$ – the ith block of the plaintext
$\boldsymbol{B}[\boldsymbol{i}][\boldsymbol{j}]$ – jth bit of iith plaintext block
$\boldsymbol{T}$ – an entire ciphertext (and similar notations as above for a ciphertext's block and bit)
$\boldsymbol{TAG}$ – an authentication tag

GENERATION OF MASKS

Two values P_i and E_i are generated for $i = 1, 2, \dots, n$+2 as given below:

$$P_i = E_{K_2}(i)$$
$$E_1 = E_{K_1}(Nonce)$$
$$E_i = ROL(E_{i-1}) \quad for\ i = 2,3..,n+2$$

where $ROL(.)$ means rotating left bit by bit.
P_i can be thought of as the extension of key K_2 and E_i of key K_1.

The values computed above are used to produce masks Y_i as given below:

$$Y_i = SUBST\ (E_i \oplus P_i)\ for\ i = 1,2,..,n$$
$$M_1 = SUBST\ (E_{n+1} \oplus P_{n+1})$$
$$M_2 = SUBST\ (E_{n+2} \oplus P_{n+2})$$

where $SUBST$ means passing the value through S-Boxes as shown in the figure 1.

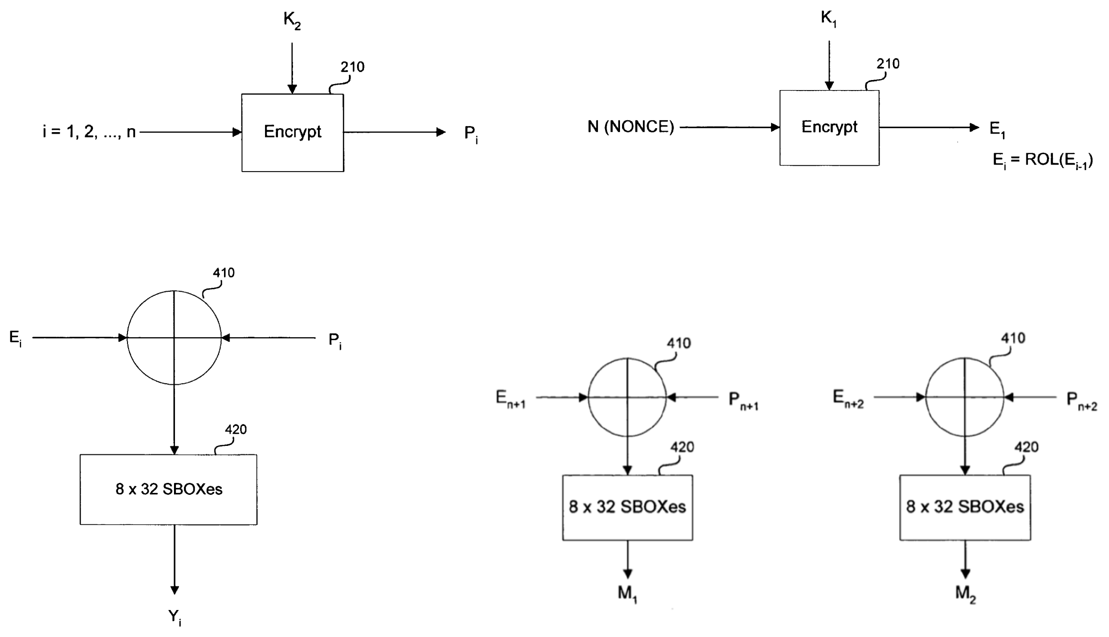

Figure 1 Mask Generation

PLAINTEXT PROCESSING

To encrypt the message, the scheme divides the messages into N blocks, where each block size is determined by the underlying block cipher. The blocks are encrypted independently and simultaneously. This makes the whole process faster.

For each plaintext block B_i of a message, the corresponding ciphertext block T_i is produced by:

$$T_i = E_{K_1}(B_i \oplus Y_i) \oplus Y_i)$$

Figure 2 below depicts the encryption process.

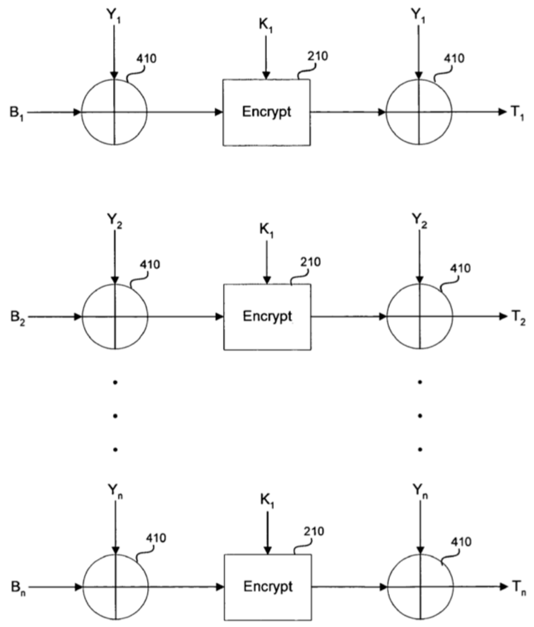

Figure 2 Encryption of plaintext

TAG GENERATION

The tag generation process makes use of the two additional tags, M1 and M2. The XOR-Sum is applied to all the message blocks in the authentication part.

The tag (denoted by "TAG") is generated using:

$$TAG = E_{K_1}(Z \oplus M_1) \oplus M_2)$$

Figure 3 depicts the authentication process.

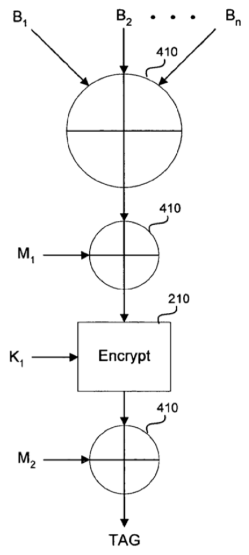

Figure 3 Authentication process

III. Security Analysis

IND-CPA & INT-CTXT

The goal of authenticated encryption schemes is to provide both privacy and authenticity. Therefore, Indistinguishability under Chosen-Plaintext Attacks (IND-CPA) and Integrity of Ciphertext (INT-CTXT) have been widely used as the security definitions under which authenticated encryption schemes are examined.

In this section, we evaluate PIAET for IND-CPA and INT-CTXT. In the previous section, we described PIAET, which is very similar to the Offset Code Book (OCB) scheme developed by Phillip Rogaway [2]. The two schemes only differ in the generation of values (using a nonce) that are used to XOR with individual plaintext blocks in a message. The overall processing structure of each plaintext block is extremely similar. Both schemes also generate a Tag for authentication by encrypting the XOR-SUM of all the plaintext blocks.

Because of this analogy, we used the results of OCB's security analysis to conclude that PIAET is IND-CPA and INT-CTXT as well.

Informally,

If there exists an adversary A that distinguishes PIAET, then there exists an adversary B that breaks the underlying block cipher E.
And similarly, if there exists an adversary A that forges a ciphertext and tag pair for PIAET, then there exists an adversary B that breaks the underlying block cipher E.

Since the overall security of the scheme depends on that of the underlying block cipher, Leech is right in suggesting the use of AES, which we know is a PRF. He also briefly mentions that

PIAET *"provides a stronger pseudo-random generator than simple Gray code used in IAPM and OCB constructions"*. This would mean that PIAET is likely to have better security bounds than OCB.

Given that PIAET is both IND-CPA and INT-CTXT, we can also conclude that it is IND-CCA.

NONCE-REUSE

IND-CCA is considered to be the strongest classical security definition under which authenticated encryption schemes are tested. Such classical models rely on strong theoretical assumptions such as a nonce being non-repeating and an IV being truly random. However, this may not be the case in a real-world setting. There have been multiple cases of faulty pseudo-random generators, some of which have compromised RSA keys. In fact, Bock et. al were able to identify 184 HTTPS servers with repeating nonces in an Internet-wide scan [3].

Therefore, it is safe to say that the security results of traditional models no longer hold in a real-world setting with weaker randomness and repeating nonces.

The PIAET scheme also relies on a receiving a non-repeating nonce as input. The nonce is the primary reason for non-determinism in the scheme. In the absence of this nonce, the same plaintext would produce the same ciphertext each time and similarly, a repeating nonce would make the scheme deterministic and it would lose both privacy and authenticity.

Leech does not put a restriction on what kind of nonce is to be used with PIAET. Using a random nonce during the transmission of massive amounts of data within the same session, increases the chances of nonce-reuse as compared to the usage of an increasing counter as a nonce.

It can be concluded that PIAET must only be used in a setting where it can be ensured that the nonce will not be repeated in a particular session. If the user of this scheme is not sure about the settings this scheme is placed in, then it is possible that the scheme does not achieve IND-CPA and INT-CTXT that it is known to achieve in theory.

INT-RUP

In our search for security models that take into account more practical settings, we came across one called Integrity under releasing unverified plaintexts (INT-RUP).

INT-RUP was a model formulated by Andreeva et. al in 2014 [4]. Given that this is not a widely used security definition, we provide a brief description of the model and also discuss why we chose to examine our scheme with it. In traditional security models, the verification oracle must first compute a plaintext from the ciphertext it received. The oracle only returns 1 if the computed plaintext is correctly verified - for example, when the Tag the oracle received matches the Tag it generated using the computed plaintext - by the verification oracle. If it is not correctly verified, then the oracle simply outputs an error. However, the release of verified plaintexts alone is more of an ideal case.

Since we did not find an intuitive model of INT-RUP, we provide one in figure 4.

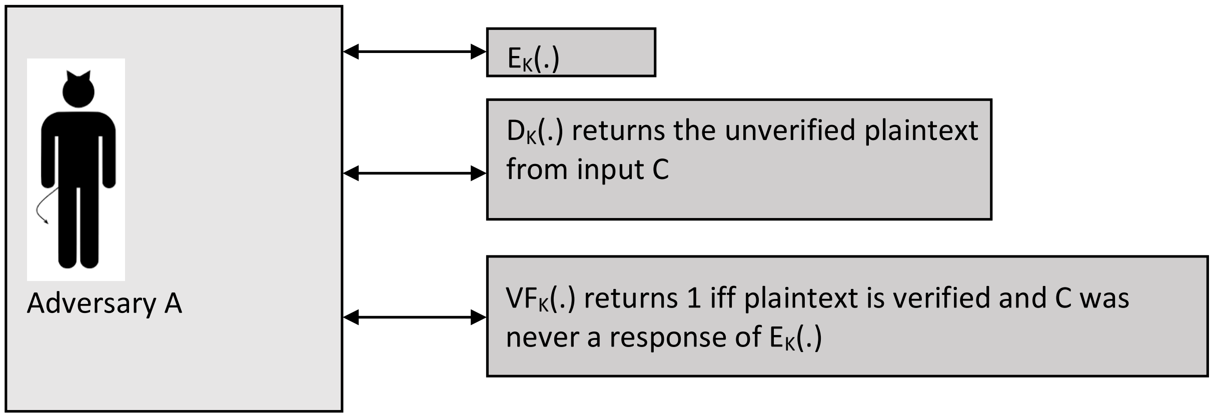
Adversary A
$E_K(.)$
$D_K(.)$ returns the unverified plaintext from input C
$VF_K(.)$ returns 1 iff plaintext is verified and C was never a response of $E_K(.)$

Figure 4 INT-RUP attack intuition

The goal of the adversary is to compute a forgery such that it returns a 1 from the verification oracle $VF_K(.)$.
The INT-RUP advantage of A is:

$$\begin{aligned} Adv^{INT-RUP}(A) &= \Pr[A\ gets\ a\ 1\ from\ VF_K\ oracle] \\ &= \Pr\,[A\ forges\ a\ valid\ ciphertext\ and\ tag\ pair] \end{aligned}$$

In many real-life situations, it is often desirable that a plaintext be released before the verification is complete. This is often seen in lightweight and small devices which may not have enough memory to store the entire plaintext or where efficiency has the highest priority. Even when unverified plaintexts are not overtly released, side channel attacks are not uncommon and sometimes extremely successful in deducing information about these plaintexts. Given the rapid rise of "Internet of Things" devices (which also tend to have low memory and at the same time are pushed to be as efficient as possible), it becomes even more important to examine schemes under the RUP setting.

We were quite impressed by this model's ability to account for various real-life situations, that are especially relevant to today's time, and thus decided to examine the PIAET scheme in the RUP setting as well.

We now provide a full INT-RUP attack on PIAET below. This attack strategy was adopted from [4] and tailored to fit our scheme.

```
1. $\boldsymbol{Adversary\ A^{E_K(.)D_K(.)VF_K(.)}}$:
   $Choose\ l\ \geq n\ where\ l = number\ of\ blocks\ and\ n = number\ of\ bits\ in\ a\ block$
   $B \leftarrow B[0]||B[1]||..||B[l]$
   $N||T||TAG\ \leftarrow\ E_K(B)\ where\ N\ represents\ the\ nonce$
   $T_0 \leftarrow any\ arbitrary\ ciphertext\ with\ l\ blocks$
   $T_1 \leftarrow any\ arbitrary\ ciphertext\ with\ l\ blocks\ such\ that\ for\ all\ i = 1,2..,l,$
                                   $each\ block\ T_2[i] \neq T_1[i]$
   $B_0 \leftarrow D_K(N, T_0, TAG_0)\ where\ TAG_0\ is\ a\ randomly\ chosen\ tag$
```

$B_1 \leftarrow D_K(N, T_1, TAG_1)\ where\ TAG_1\ is\ a\ randomly\ chosen\ tag$
$Z \leftarrow B[0] \oplus B[1] .. \oplus B[l]$

The adversary forms an equation:
$Z = (B_0[1]x_1 \oplus B_1[1]\overline{x}_1) \oplus (B_0[2]x_2 \oplus B_1[2]\overline{x}_2) .. \oplus (B_0[l]x_l \oplus B_1[l]\overline{x}_l)$
$The\ goal\ is\ to\ find\ the\ values\ x_1, x_2.., x_l\ \in GF(2).$

The above equation expands to:
$Z[j] = (B_0[1][j]x_1 \oplus B_1[1][j]\overline{x}_1) \oplus (B_0[2][j]x_2 \oplus B_1[2][j]\overline{x}_2)..$
$\oplus (B_0[n][j]x_l \oplus B_1[n][j]\overline{x}_l)\ for\ j = 0\ to\ n-1$

In the above equation, $Z[j]$ represents the j^{th} bit in Z.

The equation above represents a system of n linear equations with l unknowns in $GF(2)$ which can be solved using Gaussian elimination.

$The\ possible\ values\ of\ x_i\ are\ 0\ and\ 1. If\ x_i = 1, then\ B_0[i]\ is\ selected\ as\ the\ i^{th} message\ block\ of\ B'\ and\ if\ x_i = 0, then\ B_1[i]\ is\ selected\ as\ the\ i^{th} message\ block\ of\ B'.$

$T' \leftarrow T_{x_1}[1] || T_{x_2}[2] || .. || T_{x_l}[l]\ for\ the\ x_i\ values\ computed\ above.$
$If\ x_i = 1, then\ T_0[i]\ is\ selected\ as\ the\ i^{th} message\ block\ of\ T'\ and\ if\ x_i = 0,$
$then\ T_1[i]\ is\ selected\ as\ the\ i^{th} message\ block\ of\ T'.$

$return\ (N||T'||TAG)$

The primary reason that this attack works is because the XOR sum of the plaintext blocks is used to compute the authentication tag. It is easy for an attacker that sees two unverified plaintexts computed using the same nonce that was first used to encrypt a message, to compute another message that produces the same XOR sum, and thus the same authentication tag using that nonce.

2. $Adv^{INT-RUP}(A) = \Pr[A\ forges\ a\ valid\ ciphertext\ and\ tag\ pair]$
   $The\ probability\ of\ the\ linear\ equations\ having\ a\ solution\ by\ Gaussian\ elimination\ is\ atleast\ 1 - 2^{n-l}.$
   $Therefore,\ \Pr[A\ forges\ a\ valid\ ciphertext\ and\ tag\ pair] \geq\ 1 - 2^{n-l}$

$$\boldsymbol{Adv_{PIAET}^{INT-RUP}(A) \geq 1 - 2^{n-l}}$$

3. $Resources$:
   $t_A = time\ to\ compute\ the\ solution\ of\ the\ linear\ equations\ (polynomial)$
   $q_A = q_e +\ q_d = 1 + 2 = 3$
   $\mu_A =\ 3nl$

IV. Conclusion

In conclusion, the Parallelizable Integrity-Aware Encryption Scheme is IND-CPA and INT-CTXT when used in the right settings, that is, where the nonce is never repeated in a session. The scheme is fragile in case of nonce-reuse and the security of the scheme is completely lost. The scheme is found to be vulnerable when tested against a newer, practical security definition, INT-RUP. Additionally, a patent often discourages wide usage of a scheme even if it provides a good security guarantee.

V. Bibliography

[1] M. Leech, "Parallelizable Integrity Aware Encryption Technique". United States of America Patent US 2005/0175175A1, 11 August 2005.

[2] P. Rogaway, M. Bellare and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption," in *ACM Conference on Computer and Communications Security (CCS-8)*, 2001.

[3] H. Bock, A. Zauner, S. Devlin, J. Somorovsky and P. Jovanovic, "Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS," 2016.

[4] E. Andreeva, A. Bogdanov, A. Luykx, B. Mennink, N. Mouha and K. Yasuda, "How to Securely Release Unverified Plaintext in Authenticated Encryption," in *Advances in Cryptology – ASIACRYPT 2014.*, 2014.