Cryptographic analysis of P2DPI

Last change: 15.1.2022

Mihai-Alexandru Bogatu
{mihai.bogatu@stud.acs.upb.ro}
Politehnica University of Bucharest,
Faculty of Automatic Control and Computer Science

Abstract. This paper represents an analysis of the recently proposed P2DPI encryption scheme for deep packet inspection. Later in the article, some constructs of the algorithm are challenged and it is observed that the algorithm in this state suffers from serious integrity risks. Later in the article, there is proof that P2DPI can compromise user privacy, unlike their claims.

1. Introduction

In the context of increasing encrypted channel communication and at the same time the number of cybersecurity threats, the need for deep packet inspection is at rise. On the other hand, Intrusion Detection System/Intrusion Prevention Systems (IDS/IPS) providers wish to hide the detection rules that identify threats to minimize the possibility of an evasion, as leakage of the rules may result in clever modifications of packets to bypass the implemented security systems.

The focus of this paper is to analyze “P2DPI: Practical and Privacy-Preserving Deep Packet Inspection” proposed recently by Jongkil K. et al. This paper identifies multiple weaknesses in the P2DPI algorithm, including a compromise using exhaustive message search and proposes some countermeasures to integrity and identify proof that miss in certain places.

2. P2DPI system

2.1 Definitions

The P2DPI system is based on the property of key-homomorphic functions. A well documented premises on the usage of key-homomorphism but also detailed definitions of the cryptographic primitives used in the algorithm can be found in the original paper [1] this article analyzes. The author highly encourages reading the mentioned article before hopping into the sections detailing the attacks. The next sections use definitions based on the paper without detailing or proving the underling terms or motivations.

The parties involved in the system:

The **Rule Generator (RG)** refers to the system responsible of generating IDS/IPS rules. In most of the cases RG is a third party to an organization network. The generated rules must be kept secret from other parties with the exception of the MiddleBox. The RG should not be able to decrypt traffic or otherwise guess plain-texts from S/R.

MiddleBox (MB) is the actual system that must implement the IDS/IPS detection phase. It receives rules from RG and matches an obfuscated version of them against obfuscated tokens generated from the traffic that comes from S/R. In the case of a match, MB should take actions or alert other entities. MB should not be able to see the plain-text traffic (or otherwise decrypt the traffic) from S to R. The MiddleBox also should not be able to create its own rules.

The **Sender (S)** and **Receiver (R)** are parties interested in communicating with each-other over a private channel. S/R should not be able to see the plain-text rules. In the case one of them is compromised, by inspecting the rules, a malicious actor may be able to evade the detection systems. The RG entity may also want to preserve its intellectual property.

Notations used in the system:

- G_p finite cyclic group of prime order p. This paper considers that G_p is constructed from the generator f
- $H_1(x)$ programmable random oracle used in rule/token initial obfuscation
- $H_2(c_i, x)$ programmable random oracle that obfuscates a rule based on a counter
- k_{MB} key shared securely from RG to MB
- k_{SR} key shared securely from S to R for IDS/IPS, with no connection to the TLS communication
- r_i plain-text rule
- R_i obfuscated rule
- $sig_{RG}(R_i)$ secure signature of R_i, signed by RG
- I_i intermediate obfuscated rule for session
- S_i obfuscated session rule
- t_i plain-text traffic token
- T_i obfuscated traffic token

2.1 P2DPI algorithm

P2DPI [1] authors propose that S and R have a channel such as TLS that is used for the actual communication and each of them have a connection to MB that is used for rules matching. The TLS channel will be ignored in this paper and all traffic will refer to traffic intended for the P2DPI algorithm.

A summary of the algorithm can be found below, adapted from the original paper.

I. Setup:

1. RG generates $g, h \in G_p$ and k_{MB}, then shares them with MB
2. For each detection rule r_i, RG generates R_i and $sig_{RG}(R_i)$ and shares them with MB. R_i is computed as follows:
$$R_i = (g^{H_1(ri)} h)^{k_{MB}}$$
3. S and R start a connection with MB
4. MB send R_i and $sig_{RG}(R_i)$ that is verified by S/R. If a signature fails, output $\perp$.
5. S/R compute intermediate rules I_i and sends them to MB. The intermediary rules are calculated as:
$$I_i = R_i^{k_{SR}}$$
6. If the values from S and R I_i do not match, MB outputs $\perp$. Otherwise MB computes the session rules S_i as follows:
$$S_i = Ii^{1/k_{MB}} = (g^{H_1(ri)} h)^{k_{SR}}$$

II. Token generation:

S/R must generate tokens form the traffic for the algorithm to work. The authors of P2DPI proposed 2 methods. One of the method, used in their performance tests is as follows:

For each message m of length k bytes, make tokens of length b bytes where b is agreed on both S/R and MB/RG. Split the message m in bytes $m[0], m[1], \ldots, m[k-1]$. There must be agreed upon using padding for messages on the first and last bytes or to make tokens only from the message bytes. If no padding is applied, the tokens are formed as:

$$t_0 = m[0] || m[1] || \ldots || m[b]$$
$$t_1 = m[1] || m[2] || \ldots || m[b+1]$$
$$\ldots$$
$$t_{k-b-1} = m[k-b-1] || m[k-b] || \ldots || m[k-1]$$

For experimental tests, the authors of P2DPI used 8 byte tokens.

III. Detection:

1. S and R generates the tokens t_i from the traffic they decrypt from one another and computes a random counter c. With that information they compute the obfuscated traffic tokens T_i. Suppose S is a leader. S will send the encrypted tokens to MB and R, where T_i is equal to:
$$T_i = H_2(c+i, (g^{H_1(t_i)} h)^{k_{SR}})$$
2. The other host R verifies the obfuscated token T_i by using its own generated token. If it is a mismatch, notify MB.
3. MB compares the obfuscated traffic tokens T_i with each session obfuscated rule $H_2(c+i, S_j)$ for j in the number of session rules.

3. Attacks against P2DPI

This section of the article presents attacks discovered against P2DPI. Section 3.1 tackles the issue that only RG can know the plain-text rules. Section 3.1 presents the issues with insufficient data integrity checks and too-permissive trust. Section 3.3 presents an attack on the algebra formulation that gives room to exhaustive message search attack. Section 3.4 turns the attack from the previous section into a complete session decryption attack, proving a total compromise of user privacy.

3.1 Rules are only known by the Rule Generator

Algorithm 1 (page 13) [1] mentions that RG generates $g, h \in G_p$ and k_{MB}, then shares them with MB. Afterwards it generates the encrypted rules Ri and their signature $sig_{RG}(R_i)$. MB however can only confirm that the rules come from RG by the signature, but it can not check if the rules have malicious intentions. MB can not get r_i from R_i as $H_1(r_i)$ is a random oracle and also finding $H_1(r_i)$ from $g^{H_1(r_i)}$ represents a hard computational problem. As such, S/R and MB can agree on the number of rules, however they can not know what the rules contains.

This flaw allows RG to forge any kind of rule without any other party being able to verify it's intentions. Thus, RG can forge rules intended to compromise S/R privacy, limited only to the number of rules the other parties agree to accept. However, this alone is not enough to compromise the encryption scheme. Even if RG generates a low entropy data that it wants to match against S/R obfuscated tokens, it will not receive any information as long as MB does not exfiltrate alerts to RG.

A more secure algorithm would be to send r_i', an encrypted rule with a key MB also possesses, and $sig_{RG}(R_i)$ to MB. MB will then compute R_i by first decrypting r_i' and check against $sig_{RG}(R_i)$. That way, MB can inspect for ill-intended rules in the case of the RG threat.

3.2 Insufficient signatures/certificates

Sub-section 3.2.1-3.2.3 present vulnerabilities related to insufficient signatures/certificates from different angles. Section 3.2.4 proposes some countermeasures to the attacks.

3.2.1 Middlebox identity

By not certifying MB's messages, an attacker can use Man-in-the-middle (MITM) attack to impersonate MB. The rules are however encrypted and signed by RG. If the signature

algorithm is secure, this attack has no benefits to an ordinary eavesdropper since P2DPI uses counter-based encryption of the messages. However, if the threat actor is RG having MITM capabilities, it can essentially cut the MB from the encryption scheme as it contains all the secrets MB has for the encrypted communication.

This would mean that even if MB would be able to know the rules r_i and deny them based on their intentions, a MITM rule generator threat could still send the rules to S/R and intercept the encrypted tokens. Since S/R can not decrypt the rules by any means (as this is intended in the definition of security proposed by P2DPI), they would have to trust MB. However in this case MB can be subtracted by the encryption scheme in a MITM attack due to missing a certificate and signature. Further combined with the reasons mentioned in the 3.1 section, MB can not decrypt the traffic either, so even if it had a certificate, S/R and MB must still blindly trust the rules from RG.

This vulnerability will be further chained with an other issue to an effective privacy compromise for S and R.

3.2.2 Corruption of intermediary encrypted rules

P2DPI lacks validation of messages from S/R designed to be used in the IDS/IPS system.

Since data from S/R is not securely validated (ex. using a trusted certificate chain) in the setup process, an MITM adversary can corrupt the messages that come from S and R intended for MB containing the intermediate encrypted tokens I_i. Consider an adversary positioned both in between S-MB and R-MB or positioned in one of the mentioned connections and fully controlling the other host (R-MB and controlling S or S-MB and controlling R). After corrupting the intermediate obfuscated tokens, the session tokens will also be malformed. Any attempt to match a token with a rule will fail as their original message will not be the same, as such, a MITM attack in the setup process fully compromises any further matching attempt.

Details on the impact of the attack can be found in the section 3.2.4.

3.2.3 Corruption of validation phase or replay-attacks

As in the section before, P2DPI lacks validation of messages from S/R designed to be used in the IDS/IPS system. The algorithm proposed to defend against random obfuscated traffic tokens has vulnerabilities as well.

An other problem comes from not validating the integrity of tokens and counter (c,Ti) from R/S. In this case the algorithm is different, however, even less secure. In this case only S sends the obfuscated tokens *Ti* and the counter *c*. R verifies them and warns MB if a mismatch is made. R is considered trusted, however the following attack applies in the reversed role of R and S as well. An adversary sets a MITM attack on the traffic MB-R and

either controls S or has a MITM attack on S-MB. The attacker changes the counter the tokens values to something random (in the case of tokens, the random value should be chosen from values located on the cyclic group). The attacker then captures the traffic from R. If MB just waits passively a warning, the adversary could just drop all warning requests before they reach MB. If however MB waits actively for warnings, requesting the status of any mismatch, the adversary could just drop a warning from R and send a reply-attack with a clean status. Since there is no mention of a counter-mode based encryption with secure signatures of the alert to MB, this attack is possible.

3.2.4 Countermeasures to integrity compromise

To circumvent the risk mentioned in section 3.2.1, MB should also compute the signature of R_i through a secure signing algorithm and send it alongside the signature from RG to the clients S/R. RG should not possess the key to sign messages in the name of MB. In this case S/R can verify that the obfuscated rules Ri are in-fact from RG and they are verified by MB.

Sections 3.2.2 and 3.2.3 requires that all three parties: MB, S and R sign their messages. MB will be able to verify S and R messages integrity and S/R verify MB messages integrity. However in the case of 3.2.3 the recommendation is for both parties to send the obfuscated tokens as having a one-sided validation on the hosts is not the best practice and should be done centralized at the MB. In any case, MB should be the one raising alerts in a network-based IDS/IPS environment anyways.

While the MITM on both sides is difficult to achieve, in practice the server-client model could suffer a compromise such that an attacker obtains access to an internal IP from the server's network. Gaining MITM access to the server from the internal network is hard but not necessary impossible and in the case an adversary even with an IDS in place. There are cases when a security system is implemented after an attacker gets a backdoor to the internal network. Other attack vectors include an internal employee threat.

P2DPI system supposes that 2 from hosts that have a connection through MB, at least one is honest, otherwise both of them could just encrypt the traffic one more layer and the conversations can not be inspected. However in this state of the algorithm, if there are 2 malicious hosts on the sides of MB, all other hosts on either part of MB will be compromised without an IDS/IPS alert. This attack scenario should be taken in consideration for future privacy-oriented deep packet inspection solutions as in practice, compromise of two hosts should not compromise the entire network capabilities to detect intrusions.

3.3 Exhaustive message search vulnerability

Suppose $g, h \in G_p$ where G_p is a finite cyclic group of prime order *p* generated by *f*. Under this consideration, the following equations apply: $g^{\alpha} \equiv h^{\beta} \equiv f$, where α and β are integers in the domain *(1,2,…, # $E(G_p)$))* where $\# E(G_p)$ is the order of G_p .

Under this assumption, the key-homomorphic equation from P2DPI scheme can be rewritten as follows, considering (Abelian) finite cyclic groups proprieties:

$$H(x,k) = (g^{H_1(x)} h)^k \equiv ((f^{\alpha})^{H_1(x)} f^{\beta})^k = (f^{\alpha H_1(x) + \beta})^k$$

Notice that in the page 13 of [1] there is mentioned that RG is responsible for generating $g, h \in G_p$, this means it has access to the parameters α and β. The choice of using two functions instead of one is mentioned in Annex 5 of page 11 [1], as the simplified algorithm was used in a previous version of the article [2] but changed to this equation "due to the malleability of the previous choice". However this article demonstrates that RG can in fact reduce the problem to the one before, furthermore it can be exploited to compromise the users privacy.

With access to those parameters, the user confidentiality can be broken as follows:

<u>The attack:</u>

1. RG sends to S/R the obfuscated rule R_x and its signature.
$$R_x = (g^{H_1(x)} h)^{k_{MB}}$$
2. S/R verifies the signature of the rule and computes the intermediate obfuscated rule and sends them to MB.
$$I_x = R_{rg}^{k_{SR}}$$
3. RG computes the session reduction rules
$$S_x = I_x^{1/k_{MB}} = (g^{H_1(x)} h)^{k_{SR}}$$
4. Compute the generator encrypted with k_{SR}
$$f^{k_{SR}} = (S_x)^{1/(\alpha H_1(x) + \beta)}$$
5. Apply the parameters individually to get a function that can generate any message encrypted with k_{SR}
$$S_{rg} = (f^{k_{SR}})^{\alpha} = f^{\alpha k_{SR}} = g^{k_{SR}}$$
$$S_{rh} = (f^{k_{SR}})^{\beta} = f^{\beta k_{SR}} = h^{k_{SR}}$$
$$S(msg) = (S_{rg})^{H_1(msg)} S_{rh} = (g^{k_{SR}})^{H_1(msg)} h^{k_{SR}}$$
And by using the commutativity property of the group, can be rewritten as:
$$S(msg) = (g^{H_1(msg)} h)^{k_{SR}}$$
This means that RG can now encrypt any message with the key k_{SR} without actually knowing it.
6. For each token and counter coming from S/R, store them and apply exhaustive message search on the function
$$T_{msg} = H_2(c + i, S(msg))$$
compared with

$$T_i = H_2(c+i, S_i)$$

When a match is found, the adversary guessed a message.

By allowing RG to generate g, h, it can effectively compromise the users privacy.

Lets consider that the parameters g, h are generated by a third party and the private keys that make g, h from f are not shared with RG.

The encryption scheme makes it such S/R can choose the encrypted rules they want to accept from MB, signed by RG. However the contents of the rules are encrypted and as such they can't know for sure that the rule contains. This was a basis of the algorithm as knowing or obtaining a decryption by any means of a plain-text rule on any other party than RG or MB compromises their privacy. One issue in P2DPI, is that the middlebox can not check the rules. As such a malicious RG can generate any kind of rule, limited to the number of rules MB and S/R agree to accept. This was pointed out in section 3.2.1 of this article. However, the privacy of S/R is the subject of interest for both MB and RG and as such they may cooperate to compromise S/R privacy anyways.

In this context, this article will prove that P2DPI is vulnerable to exhaustive message search.

Let ϕ be the neutral element in the ring addition of the cyclic group G_p. As an example, in Elliptic Curve Cryptography (ECC), ϕ is the point at infinity. Under a finite cyclic group of prime order, the neutral element is present in any closed group.

By the definition, for any $f \in G_p$ with an inverse in a closed group following property applies:

$$f * f^{-1} = \phi$$

As such, consider the key σ the smallest non-zero value for which $f^{\sigma} = \phi$. One example is to set $\sigma = p-1$ where p is the prime order of the group G_p by using Fermat's Little Theorem $f^{p-1} \equiv 1 \, mod \, p$, where 1 is the neutral element by definition.

By setting $g^{H_1(x)} = f^{\sigma}$ RG can generate the following payload, taking in the consideration the commutativity property of the group:

$$R_r = (g^{H_1(x)} h)^{k_{MB}} = (f^{\sigma} f^{\beta})^{k_{MB}} = (\phi f^{\beta})^{k_{MB}} \equiv (f^{\beta})^{k_{MB}} = h^{k_{MB}}$$

Note that $g^{H_1(x)} = \phi$ is a valid element in the group. However it is hard to find x such that $H_1(x) = \sigma$ but at the same time, for S/R it is hard to prove that RG actually calculated the payload. For that reason an adversary can set $g^{H_1(x)} = \phi$ without computing the actual x that satisfies the equation. The same is true for $h^{k_{MB}} = \phi$. In practice, RG can just send $g^{k_{MB}}$ and $h^{k_{MB}}$ to be signed by S/R without making the actual computations.

The attack would be harder to employ if S/R could check if the obfuscated rules come from the same key used in generating the rules, as it should be.

For the attack to work with only 2 rules in that scenario, the following mathematical system must be valid in Z_p (integers modulo p), for the signature to match:

$$\begin{cases} (\alpha x_1 + \beta) k = \alpha * k_1 \\ (\alpha x_2 + \beta) k = \beta * k_2 \end{cases}$$

Where x_1, x_2, k_1, k_2 are chosen arbitrary from Z_p such that α and β can take any value, besides the null element, and the equation would still apply.

We can choose k_1, k_2 only if x_1, x_2 exist Z_p under that equation. A necessary condition is that α is a multiple of β. The idea behind is that an attacker with a private key can apply multiple rules and form a basis that under some additions and multiplications will end up forming g and h and thus bypass the validation, without having α and β.

Below is a detail of the attack using 2 rules without taking in consideration the signature proof.

The attack by multiple rules:

1. RG sends to S/R the obfuscated reduction rules R_{rg}, R_{rh} and their signatures.
$$R_{rg} = g^{k_{MB}}$$
$$R_{rh} = h^{k_{MB}}$$
2. S/R verifies the signature of the rule and computes the intermediate obfuscated rule and sends them to MB. Since S/R do not hold the key k_{MB}, it's hard to prove that R_{gh} and R_{rh} come from exclusivity g and h as they are valid rules anyways.
$$I_{rg} = R_{rg}^{k_{SR}} = (g^{k_{MB}})^{k_{SR}} = g^{k_{MB} k_{SR}}$$
$$I_{rh} = R_{rh}^{k_{SR}} = (h^{k_{MB}})^{k_{SR}} = h^{k_{MB} k_{SR}}$$
3. RG computes the session reduction rules
$$S_{rg} = I_{rg}^{1/k_{MB}} = g^{k_{SR}}$$
$$S_{rh} = I_{rh}^{1/k_{MB}} = h^{k_{SR}}$$
4. Having both $g^{k_{SR}}$ and $h^{k_{SR}}$, RG can now forge any obfuscation of a message m by computing:
$$S(msg) = (S_{rg})^{H_1(msg)} S_{rh} = (g^{k_{SR}})^{H_1(msg)} h^{k_{SR}}$$
Which, due to commutativity property of the group, can be rewritten as:
$$S(msg) = (g^{H_1(msg)} h)^{k_{SR}}$$
This means that RG can now encrypt any message with the key k_{SR} without actually knowing it.
5. For each token and counter coming from S/R, store them and apply exhaustive message search on the function
$$T_{msg} = H_2(c + i, S(msg))$$
compared with
$$T_i = H_2(c + i, S_i)$$
When a match is found, the adversary guessed a message.

Note that 2 random parameters φ, ψ could be chosen by RG to send R_{rg}^{φ} and R_{rh}^{ψ}. RG can still compute $S_{rg} = (S_{rg}')^{1/\varphi}$ and $S_{rh} = (S_{rh}')^{1/\psi}$ since the values were generated by the attacker. This would further obfuscate RG's intentions to the already oblivious S/R, however the parameters φ, ψ must take the obfuscated rules in the domain space generated of the

original obfuscation function. While for R_{rh} always exists on the domain as there will always be a value for $H_1(x)=\sigma$, one must find a value of φ to make sure that R_{rg}^{φ} is included in the domain of values R outputs.

This attack is realistic since MB does not sign the messages and even if it would sign them, MB does not require from RG the plain-text rules. As such, this attack demonstrates that P2DPI is vulnerable to exhaustive message search attack. This would have a big impact on low-entropy data, however by using a counter at each token, the risk is minimized. On the other hand, $H_2(ci,msg)$ is not modeled as a slow function, as it should not be because it is used in the algorithm extensively for real-time data coming from the traffic tokens. In this case, a brute force on H_2 based on the index is possible on a realistic time to find a collision, as long as the token size is small.

An attacker has to do 2^{64} comparisons per token for an 100% chance of success in the case of truly random plain-text data with tokens with a length of 8 characters (as used in the demonstration section of P2DPI algorithm). However, since protocols are predictable, and in most situations data transferred is composed of printable characters or otherwise encoded with a transformation of the data in a printable format (such as base64), the attack complexity is reduced further. For example, for a complete ASCII encoding that uses 7 bits of data out of a byte, an attack would be halved, reducing the complexity to 2^{32}, which can be reduced further with other techniques.

In this section, of this paper demonstrated attacks where only one of the directly involved member is malicious.

However the authors of P2DPI pointed on the section 2.2 Threat Model that a collaborating MB and RG could not compromise S/R confidentiality more than S/R agrees to. However by using 2 specially crafted rules, the author of this article demonstrates that MB and RG could in fact compromise the entire confidentiality of S and R sessions by getting a single token from the session, under the presumptions of the attack. As such even if sufficient integrity checks are taken in place, this attack would still break S/R confidentiality if MB/RG collaborate.

3.4 Byte-at-a-time attack

Under the presumptions of the previous section, that RG with MITM capabilities or RG with the collaboration of MB could generate the algorithm reduction parameters and construct $S(msg)=(g^{H_1(msg)}h)^{k_{SR}}$ without the knowledge of the key k_{SR}, this section demonstrates that in the case of a token generation method used by P2DPI and mentioned in BlindBox [3] as well, an adversary controlling the mentioned possible threats obtaining a single token plain-text could compromise the entire confidentiality of S/R.

Lets assume that MB/RG obtained the token t_i. Since t_{i-1} and t_{i+1} differ by a singe byte, obtaining the adjective tokens by knowing their obfuscation T_{i-1}, T_{i+1} and having

access to the encryption algorithm that feeds before $H_2(ci, S_i)$ as in the form of encryption oracle, the attack is no different than an byte-at-a-time ECB attack.

For t_{i+1}, the attacker sets $t_{i+1}[0:b-1] := t_i[1:b]$ with where b is the token size. The last byte of t_{i+1} is then brute-forced until $H_2(ci, S_i) = H_2(ci, S(t_i))$. When a match is found, then we obtained the token t_{i+1}. This can be repeated for each byte of the message. The attack can be used to get the previous bytes as well.

This attack concludes that a single guessed token from the session compromises the confidentiality of all the other bytes of the message. This can be done easily on predictable tokens such as by targeting headers in HTTP protocol.

4. Definition Middle Box Searchable Encryption deficit

Although the proofs of the definitions of MSBE security present in P2DPI are correct, the actual definitions of MSBE security present on P2DPI article are incomplete.

Below are listed the definitions of MSBE security as found in the article.

Definition 4 (MBSE) *Middlebox searchable encryption consists of five algorithms, namely* Setup, RuleEnc, Encrypt *and* Match*:*

- Setup $[1^\lambda \rightarrow (sk, pk)]$ accepts a security parameter 1^λ and outputs an encryption key sk and a public parameter pk.
- RuleEnc $[(sk,\ r,\ pk) \rightarrow\ S)]$ accepts sk and r from a message space $\mathcal{M}$, and outputs an obfuscated rule S.
- Encrypt $[(sk, \mathcal{T}, pk) \rightarrow\ (param, \mathcal{ET})]$ accepts sk and a set of tokens $\mathcal{T} = \{t_1, ..., t_n\}$ from $\mathcal{M}$ for $n = \text{poly}(\lambda)$ and outputs encryption parameters $param$ and encrypted tokens $\mathcal{ET} = \{T_1, ..., T_n\}$.
- Match $[(param,\ \mathcal{ET},\ S) \rightarrow\ \mathcal{I}]$ accepts $param$, $\mathcal{ET}$ and S. It outputs the set of indices $\mathcal{I} = \{i | t_i = r\}$.

MSBE Base [1 definition 4, page 15]

Definition 5 (MBSE Security) *Let $\mathcal{A}$ be a stateful PPT adversary. Consider the following security game* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{MBSE}}(1^{\lambda})$*:*

- Init: The challenger $\mathcal{C}$ runs Setup with a security parameter 1^{λ} to generate rule encryption keys sk and public parameter pk. It sends pk to $\mathcal{A}$.
- Query: $\mathcal{A}$ queries $\mathcal{C}$ with a set of rules $\{r_1, ..., r_q\}$. $\mathcal{C}$ invokes RuleEnc with sk and returns $\{S_1, ..., S_q\}$ to $\mathcal{A}$.
- Challenge: $\mathcal{A}$ generates two sets of tokens $\mathcal{T}_0 = \{t_{0,1}, ..., t_{0,n}\}$ and $\mathcal{T}_1 = \{t_{1,1}, ..., t_{1,n}\}$ from the message space $\mathcal{M}$ such that $\forall i \in [q], r_i \notin \mathcal{T}_0 \cup \mathcal{T}_1$. It sends both to $\mathcal{C}$. $\mathcal{C}$ randomly sets $\beta \in \{0, 1\}$ and runs Encrypt for $\mathcal{T}_\beta$ to get $(param, \mathcal{ET})$. $\mathcal{C}$ returns $(param, \mathcal{ET})$ to $\mathcal{A}$.
- Guess: $\mathcal{A}$ outputs β'. If $\beta = \beta'$, $\mathcal{A}$ wins and the game outputs 1. Otherwise, the game outputs 0.

We say that MBSE is secure if for all PPT stateful adversaries $\mathcal{A}$, there exists a negligible function $negl$ such that

$$\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{MBSE}}(1^{\lambda}) = 1] = \frac{1}{2} + negl(\lambda).$$

MSBE user Privacy [1 definition 5, page 16]

The construction of P2DPI makes it hard to get a plain-text of rule r_i by having he obfuscated rule R and the key k_{MB} due to the random oracle H_1. However this does not mean that an adversary could not send to the users for session generation a specially crafted R that would otherwise be hard to compute the original function. As long as R is still a value on the cyclic group G_p , a user has no reason not to accept the rule as log as the number of rules is not excessive.

As a more complete definition, RuleEnc should accept a plain-text rule and output an obfuscated rule from the rule-generator. An additional step, GenerateSessionRules should be included, that generates the session rule based on the obfuscated rule. As such, in the definition 5, the adversary A could use the output of RuleEnc to feed an obfuscated rule to GenerateSessionRules or it may want to craft by hand a specially obfuscated rule that may be hard to generate trough RuleEnc and feed the output to GenerateSessionRules.

In the current definitions, P2DPI would preserve user privacy, however this article proved that is not the case. As such the definitions of MSBE security present in P2DPI article are incorrect and incomplete. This article proved that the theory present in P2DPI that promises privacy should not be taken in consideration as a valid theoretical proof of security under the real attack-surface of the algorithm.

4. Conclusions

P2DPI promised a high security level, however at this stage of the algorithm, even tough it is faster than the competing methods to achieve privacy-based deep-packed inspection, the security level provided at this stage is inadequate for use at is breaks searchable encryption promise of confidentiality for the hosts that want to communicate trough a private channel. As such its privacy level is equivalent to an intercepting-proxy, however since it lacks enough message integrity proofs, it's efficiency in detecting intrusions can be compromised by having an intercepting attacker on both sides of the MiddleBox.

5. References

[1] Jongkil Kim, et al. P2DPI: practical and privacy-preserving deep packet inspection, https://eprint.iacr.org/2021/789.pdf, Accessed: December 23'th, 2021

[2] Jongkil Kim, et al. P2DPI: practical and privacy-preserving deep packet inspection, ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Virtual Event, Hong Kong, June 7-11, 2021, pages 135–146. ACM, 2021. Cross-reference from [1]

[3] Justine S., et al. Blindbox: Deep packet inspection over encrypted traffic. In ACM SIGCOMM 2015, London, United Kingdom, August 17-21, 2015, pages 213–226. ACM, 2015