Cryptographic Analysis of P2DPI

Mihai-Alexandru BOGATU
POLITEHNICA University of Bucharest, Faculty of Automatic Control and Computer Science
mihai.bogatu@stud.acs.upb.ro, alex.bogatu@yahoo.com

Abstract: This paper represents an analysis of the recently proposed P2DPI encryption scheme for deep packet inspection. Constructs of the algorithm are challenged with a complete confidentiality and integrity evaluation. Later in the article, there is proof that, in the current form, usage of P2DPI can compromise user privacy.

Keywords: Cryptography, Exhaustive Message Search, Key-Homomorphic Encryption

INTRODUCTION

In the context of increasing encrypted channel communication and at the same time of 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. In the current technological age, there is a battle between evasion techniques and the detection accuracy, in the realm of cybersecurity.

The focus of this paper is to analyze "P2DPI: Practical and Privacy-Preserving Deep Packet Inspection" proposed recently by Kim et al. (2021a). This paper identifies multiple weaknesses in the P2DPI algorithm, including a compromise using exhaustive message search and proposes some countermeasures to integrate and identify proof that miss in certain places.

DEEP PACKET INSPECTION WITH MUTUAL PRIVACY

Current advances in research have been made to implement deep packet inspection algorithms that allow both scanning encrypted traffic and also preserving the confidentiality of the hosts communication and recently, the detection rules themselves. Noticeable examples include: Blindbox (Sherry et al., 2015) that uses Yao's garbled circuits (Yao, 1986) in an AES obfuscation construct, PrivDPI (Ning et al., 2019) that proposed usage of cyclic groups and P2DPI (Kim et al., 2021a) that achieved better performance than PrivDPI and at the same time, addressed certain security problems regarding the previous algorithm.

DEFINITIONS

The parties involved in the system:

The ***Rule Generator (RG)*** refers to the system responsible for generating IDS/IPS rules. In most of the cases RG is a third party to an organization's 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. Also the Middlebox 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.

P2DPI ALGORITHM

The P2DPI system is based on the property of key-homomorphic functions. A well-documented premise 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 (Kim et al., 2021a) this article analyzes. The author highly encourages reading the mentioned article before hopping into the sections detailing the attacks.

P2DPI authors propose that the Sender and Receiver 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 and session rule creation. The TLS channel will be ignored in this paper and all traffic will refer to traffic intended for the P2DPI algorithm.

Notations used in the system:

The notations used in this paper are highly close to the ones used by the authors of P2DPI.

- 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.

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 an obfuscated rule R_i and its signature $sig_{RG}(R_i)$. It is then shared with MB. R_i is computed by using a programmable random oracle H_1 as follows:

$$R_i = \left(g^{H_1(r_i)} h\right)^{k_{MB}}$$

3. S and R securely share the keys k_{SR} between them and start a connection with MB. For each detection rule, MB sends R_i and $sig_{RG}(R_i)$ that is verified by S/R. If a signature fails, output $\perp$.
4. S/R compute the intermediate rules I_i and sends them to MB. The intermediary rules are calculated as:

$$I_i = R_i^{k_{SR}}$$

5. If the values from S and R I_i do not match, MB outputs $\perp$. Otherwise MB computes the session rules S_i and reuses them in traffic matching as follows:

$$S_i = I_i^{1/k_{MB}} = \left(g^{H_1(r_i)} h\right)^{k_{SR}}$$

II. Token generation:

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

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 an agreement concerning the usage of padding for messages on the first and last bytes or the making of 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.

Note that the rules must also be made into tokens. In the case a rule is smaller than the window size, all possible combinations of the rule and the rest of the window size must be computed, with regards to the protocol that is covered by the rule. If a rule is longer than the window size, it can be broken down in multiple tokens. Last tokens can contain redundancy starting with a specific offset to avoid generating the case of a rule smaller than the window size. The Middlebox should be instructed about rules broken into multiple tokens for alerting and efficiency purposes.

III. Detection:

In this case, the Sender is considered a master for communication. It follows the server-client model.

1. S generates the tokens t_i from the decrypted traffic that is sent through the TLS channel. A random counter c is chosen. With that information, compute the obfuscated traffic tokens T_i. S will send the counter and the encrypted tokens to MB and R, where T_i is equal to:

$$T_i = H_2\left(c+i, \left(g^{H_1(t_i)}h\right)^{k_{SR}}\right)$$

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 obfuscated session rule $H_2(c+i, S_j)$ for j in the number of session rules. If a match is found, an alert is raised.

ATTACKS AGAINST P2DPI

This section of the article presents attacks discovered against P2DPI. Section *Rules are only Known by the Rule Generator* tackles the issue that only RG can know the plain-text rules. Section *Insufficient Signatures/ Certificates* presents the issues with insufficient data integrity checks and too-permissive trust. Section *Exhaustive Message Search Vulnerability* presents an attack on the algebra formulation that gives room to exhaustive message search attack. *Byte-at-a-time attack* turns the attack from the previous section into a complete session decryption attack, proving a total compromise of user privacy.

RULES ARE ONLY KNOWN BY THE RULE GENERATOR

Algorithm 1 (Kim et al., 2021a) mentions that RG generates $g, h \in G_p$ and k_{MB}, then shares them with MB. Afterwards it generates the encrypted rules R_i and their signature $sig_{RG}(R_i)$. However MB 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 contain. This is intended for S/R. However, not giving this information to MB may raise a problem.

This flaw allows RG to forge any kind of rule without any other party being able to verify its 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 the obfuscated rules R_i with their signatures $sig_{RG}(R_i)$ 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 for the case of an RG threat that attacks the algorithm itself.

INSUFFICIENT SIGNATURES/CERTIFICATES

Sub-sections *Middlebox Identity; Corruption of Intermediary encrypted Rules* and *Corruption of Validation Phase or Replay-Attacks*, present vulnerabilities related to insufficient signatures/certificates from different angles. Section *Countermeasures to Integrity Compromise* proposes some countermeasures to the attacks.

MIDDLEBOX IDENTITY

By not certifying MB's messages, an attacker can use a Man-in-the-middle (MITM) attack to impersonate MB. However, the rules are obfuscated and signed by RG. If the signature algorithm is secure, this attack has no benefits for compromising the confidentiality more than an ordinary eavesdropper since P2DPI uses counter-based encryption of the messages. Still, 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 obfuscated 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 section before, 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 another issue identified later in the article to achieve an effective privacy compromise for S and R.

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 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. 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 (in between R-MB and controlling S or in between 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 section *Countermeasures to integrity copromise*.

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.

Another problem comes from not validating the integrity of tokens and counters (c, T_i) from R/S. In this case the algorithm is different, however, even less secure. In this case only S sends the obfuscated tokens T_i 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 in between MB-R and either controls S or has a MITM attack in between S-MB. The attacker changes the chosen counter 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 for 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 for the alerts that should be send to MB, this attack is possible.

COUNTERMEASURES TO INTEGRITY COMPROMISE

To circumvent the risk mentioned in the section *Middlebox Identity*, 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 R_i are in-fact from RG and they are verified by MB.

Sections *Corruption of Intermediary encrypted Rules* and *Corruption of Validation Phase or Replay-Attacks* require that all three parties: MB, S and R sign their messages. MB will be able to verify the integrity of S and R messages and the other way around, S/R verify the integrity of MB messages. However in the case of *Corruption of Validation Phase or Replay-Attacks* 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 all the alerts in a network-based IDS/IPS environment.

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 necessarily impossible even in the case of an IDS in place. There are cases when a security system is deployed after an attacker already 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.

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: $f^{\alpha} \equiv g, f^{\beta} \equiv h$, 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 properties:

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

Notice that in (Kim et al., 2021a) there is mentioned that RG is responsible for generating $g, h \in G_p$. This means that it has access to the parameters α and β. The choice of using two functions instead of one is mentioned in (Kim et al., 2021a), as the simplified algorithm was used in a previous version of the article in Annex 5 of (Kim et al., 2021b) 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:

The attack:

1. RG sends to S/R any obfuscated rule R_x and its signature.

$$R_x = \left(g^{H_1(x)}h\right)^{k_{MB}}$$

2. S/R verifies the signature of the rule and computes the intermediate obfuscated rule and sends them to MB.

$$I_x = R_x^{k_{SR}}$$

3. RG computes the session rules:

$$S_x = I_x^{1/k_{MB}} = \left(g^{H_1(x)}h\right)^{k_{SR}}$$

4. Then, RG can compute the generator of the group encrypted with k_{SR}:

$$f^{k_{SR}} = \left(S_x\right)^{1/\left(\alpha H_1(x)+\beta\right)}$$

5. Apply the parameters individually to get a function that can generate any message encrypted with k_{SR}:

$$S_{rg} = \left(f^{k_{SR}}\right)^{\alpha} = f^{\alpha k_{SR}} = g^{k_{SR}}$$

$$S_{rh} = \left(f^{k_{SR}}\right)^{\beta} = f^{\beta k_{SR}} = h^{k_{SR}}$$

$$S(msg) = \left(S_{rg}\right)^{H_1(msg)} S_{rh} = \left(g^{k_{SR}}\right)^{H_1(msg)} h^{k_{SR}}$$

By using the commutative property of the group, the equation above can be rewritten:

$$S(msg) = \left(g^{H_1(msg)}h\right)^{k_{SR}}$$

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 by using the function:

$$T_{msg} = H_2\left(c+i, S(msg)\right)$$

and compare the results with all the tokens

$$T_i = H_2\left(c+i, S_i\right)$$

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

By allowing RG to generate g,h, it can effectively compromise the user's privacy.

Let us 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 that 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 pointed out in *Middlebox identity* of this article is that 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. However, the privacy of S/R is the subject of interest for both MB and RG. As such they may cooperate to compromise S/R privacy anyways.

In this context, this article will provide another attack to P2DPI that makes room for exhaustive message search even if the first attack is mitigated.

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 or added in any closed group for the algorithms to work.

By the definition, for any $f \in G_p$ with an inverse in a closed group, the 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 into consideration the commutative property of the group:

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

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 encrypted by S/R with their keys without making the actual computations to generate the obfuscated tokens that match those values.

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. Note that in this scenario, we consider that RG doesw not generate or know those values.

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 getting g and h, thus being able to 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 as no properties of the verification algorithm were present in the P2DPI algorithm.

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.

$$I_{rg} = R_{rg}^{k_{SR}} = \left(g^{k_{MB}}\right)^{k_{SR}} = g^{k_{MB}k_{SR}}$$
$$I_{rh} = R_{rh}^{k_{SR}} = \left(h^{k_{MB}}\right)^{k_{SR}} = h^{k_{MB}k_{SR}}$$

3. RG computes what will be called 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:

$$S(msg) = \left(S_{rg}\right)^{H_1(msg)} S_{rh} = \left(g^{k_{SR}}\right)^{H_1(msg)} h^{k_{SR}}$$

Which, due to commutative property of the group, can be rewritten as:

$$S(msg) = \left(g^{H_1(msg)}h\right)^{k_{SR}}$$

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:

$$T_{msg} = H_2\left(c + i, S(msg)\right)$$

compared with the obfuscated tokens:

$$T_i = H_2\left(c + i, S_i\right)$$

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

Note that two random parameters φ, ψ could be chosen by RG to send R_{rg}^{φ} and R_{rh}^{ψ}. RG can still compute $S_{rg} = \left(S_{rg}'\right)^{1/\varphi}$ and $S_{rh} = \left(S_{rh}'\right)^{1/\psi}$ 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 from the original obfuscation function. While for R_{rh} always exists a value 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, $H_2(c_i,msg)$ is not modeled as a slow function, as it should 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 a 100% rate 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 it is encoded with a transformation of the data in a printable format (such as base64), the attack complexity is further reduced. For example, a complete ASCII encoding that uses 7 bits of data out of a byte, would reduce the complexity to 2^{56}, which can be further reduced with techniques that are based on character frequency or other probabilistic attacks.

This section, demonstrates attacks where only one of the directly involved member is malicious.

However the authors of P2DPI pointed in section *P2DPI Algorithm*, the threat model involving a MB and RG collaboration could not compromise S/R confidentiality more than S/R agrees to. However, by using two 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.

BYTE-AT-A-TIME ATTACK

Under the previous mentioned presumptions, RG with MITM capabilities or RG with the collaboration of MB could use the algorithm mentioned in the previous section and construct $S(msg)=\left(g^{H_1(msg)}h\right)^{k_{SR}}$ 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 (Sherry et al., 2015) as well, an adversary that obtains a single token's plain-text could compromise the entire confidentiality of S/R.

Let's assume that MB/RG obtained the token t_i. Since t_{i-1} and t_{i+1} differ by a single byte from t_i, obtaining the adjacent tokens by knowing their obfuscation T_{i-1},T_{i+1} and having access to the encryption algorithm that feeds before $H_2(c_i,S_i)$ as in the form of an encryption oracle, the attack is no different than an byte-at-a-time ECB attack.

For t_{i+1}, the attacker sets $t_{i+1}[0:b-1]:=t_i[1:b]$, 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 the token t_{i+1} is obtained. This can be repeated for each byte of the message. The attack can be used to get the tokens before the known token is used as a pivot 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.

PROPOSED MUTUAL PRIVACY DEEP PACKET INSPECTION DEFINITIONS

Clear formal definitions of threat vectors should be made for the problem that was tackled by multiple algorithms in regards to privacy-based deep packet inspection. P2DPI proposed some formal definitions, however it lacked outlook on some possible problems that may appear in a possible implementation of the algorithm. As such, this section provides an introspective view of the attack surface based on the communication channels and parties involved in the system.

Definition 1, hosts privacy. Irrespective of the number of parties involved in the system, two honest hosts that wish to keep a secretive conversation, should have an adequate security level that preserves confidentiality and integrity of the messages sent, regardless of the number, type and positioning of malicious entities. The only information that could be learned about the traffic of the hosts is in the form of tokens that the hosts agree to have matched against the communication. Regardless of the number and content of agreed tokens, an entity should not be able to learn any information of the hosts communication that is not already given by the cumulative information of the accepted tokens and their matches, by using an efficient algorithm with an advantage higher than that of a non negligible function.

Definition 2, rule confidentiality. No other party than the rule generator and the rule matcher should be able to learn any information regarding a rule. The only exception is the alerting handling system that should communicate with the rule matcher, through a secure communication channel. All parties involved in generation and matching of rules must have integrity and identity controls in place.

Definition 3, robust alerting. Integrity and identity checks must be implemented at the level of communication between the hosts and the middlebox responsible with the matching of rules with the traffic. As long as there is at least one honest host, the middlebox must be able to detect active tampering of the data in transit and respond with adequate alerts.

IMPLEMENTATION

The vulnerability was implemented in a scenario where the rule generator did not have direct access to the parameters responsible for the cyclic group base of the algorithm. The implementation used Elliptic Curve Cryptography (ECC) for cyclic groups and Python programming language. Instead of using AES for the random oracle, the proof used Blake3 hashing function with a salt. The architecture is operating on a client-server design. Tests were performed on an Intel Core i5-4590 @ 4x 3.7GHz. By knowing four adjacent plain-text bytes from traffic, the attack recovered 411 bytes from the original message in under 5 minutes, using a parallel brute-force technique.

More optimizations can be employed however the scenario proves the attack.

The implementation of the server was also used in 54 hours long Capture the Flag competition as part of Bit Sentinel's contribution to DefCamp CTF 21-22, one of the oldest and largest cybersecurity CTF competitions in Central and Eastern Europe. Out of 307 teams with positive scores, 8 teams solved the challenge, out of which 7 finished in the top 10 teams.

CONCLUSIONS

P2DPI promised a high security level, however at this stage of the algorithm, even though it is faster than the competing methods to achieve privacy-based deep-packet inspection, the security level provided at this stage is inadequate for use, as is breaks searchable encryption promise of confidentiality for the hosts that want to communicate through 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.

ACKNOWLEDGMENTS

I would like to thank Andrei Avădănei for the help in publishing this article but also for the final qualitative checks. I would also like to thank Marios O. Choudary for the motivation on studying recent cryptographic papers as a faculty project.

REFERENCE LIST

Kim, J., Camtepe, S., Baek, J., Susilo, W., Pieprzyk, J. & Nepal, S. (2021a). P2DPI: Practical and Privacy-Preserving Deep Packet Inspection. Available at: <https://eprint.iacr.org/2021/789.pdf>. Last accessed: December 23, 2021.

Kim, J., Camtepe, S., Baek, J., Susilo, W., Pieprzyk, J. & Nepal, S. (2021b). P2DPI: Practical and Privacy-Preserving Deep Packet Inspection. In ASIA CCS '21: ACM Asia Conference on Computer and Communications Security, Hong Kong, June 7-11 (pp. 135–146). ACM. Cross-reference from (Kim et al., 2021a).

Ning, J., Poh, G. S., Loh, J.-C., Chia, J. & Chang, E.-C. (2019). PrivDPI: Privacy-Preserving Encrypted Traffic Inspection with Reusable Obfuscated Rules. In ACM Conference on Computer and Communications Security (CCS), London, UK, November 11-15 (pp. 1657–1670). ACM.

Sherry, J., Lan, C., Popa, R. A. & Ratnasamy, S. (2015). BlindBox: Deep Packet Inspection over Encrypted Traffic. In Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, ACM SIGCOMM 2015, London, United Kingdom, August 17-21 (pp. 213–226). ACM.

Yao, A. C.-C. (1986). How to Generate and Exchange Secrets. In Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS), (pp. 162-167). IEEE. DOI: 10.1109/SFCS.1986.25