Intrusion Detection Systems over an Encrypted Traffic: Problem and Solutions

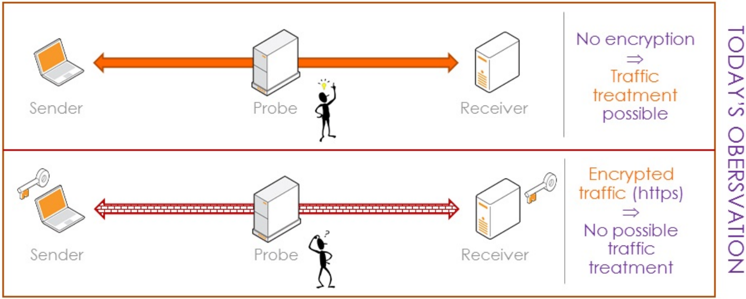
Sébastien Canard (Orange)

SoSySec seminar, December, 2019

Encryption is our future

- IETF HTTPbis working group that is in charge of designing the next generation http 2.0 specification proposes that encryption be the default way data is transferred over the open Internet
- According to a joint study by Ponemon institute, along with Thales and Vormetric Data Security, encrypted Internet traffic has grown up from 15% of world-wide traffic in 2005 until up to 40% in 2015. The proportion of encrypted Internet traffic is expected to reach up to 80% by 2020
- OTTs are moving forward towards full end-to-end encryption, including recent example such as whatsapp, Google both for end-to-end email encryption and for Internet browsing, etc.
- European Community, through its Horizon H2020 program, and in particular the joint cPPP on cybersecurity, is advocating for more privacy guarantees in terms of traffic encryption for end users
- ...

Confidentiality $\Longrightarrow$ full security?

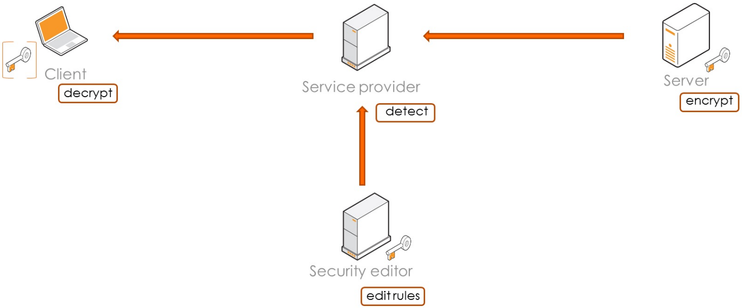


With current standards, difficult choice between
data confidentiality and usability/security!!

Impacted use cases

- Parental control over the traffic
- Security Information and Event Management
- Detecting compromising SSH requests
- Quality service probes
- Intrusion Detection Systems (IDS)
- ...

Architecture



- Deep Packet Inspection on the content of the packet
- Use detection rules to analyse the content of the traffic
  - Behavior-based detection: mostly done on meta-data that are not encrypted (CISCO approach)
  - Signature-based detection: intrusions detection using signatures

$\Longrightarrow$ How to manage an encrypted traffic?

Agenda

- Security model
- Problem and first approach
- Dealing with security-awareness
- Dealing with practicality
- Conclusion

Security model

Main procedures

- Setup(1^λ): parameters param, $\mathsf{sk}_{\mathsf{SE}}$ for Security Editor, $\mathsf{sk}_{\mathsf{SP}}$ for Service Provider, sk_{S} for Sender, and sk_{R} for Receiver
- RuleGen($\mathsf{param}, \mathsf{sk}_{\mathsf{SE}}, \mathcal{M}$): a set $\mathcal{B}$ of blinded rules
- Send($\mathsf{param}, \mathsf{sk}_{\mathsf{S}}, T = \{t_j\}_j$): an encrypted traffic E for a receiver R
- Detect($\mathsf{param}, \mathsf{sk}_{\mathsf{SP}}, E, \mathcal{B}$): a bit $b \in \{0, 1\}$, stating that the underlying traffic T is malicious ($b = 0$) or safe ($b = 1$), and some auxiliary information aux
- Receive($\mathsf{param}, \mathsf{sk}_{\mathsf{R}}, E, \mathsf{aux}$): a plain traffic T, or an error message $\perp$

Requirements and assumptions

- High level properties
  - Privacy-friendly: no access is possible to the clear-text content of encrypted traffic
  - Security-aware: it supports DPI over encrypted traffic
  - Market-compliant: it achieves real-world market requirements, including rule secrecy (know-how of the Security editor)
  - Practical: it provides good performances, both in time and memory
- Assumptions on players
  - Service Provider is honest-but-curious on both the traffic and the rules
  - Collusion between Service Provider and Security Editor cannot be handled, due to dictionary attack
  - Collusion between Client and Server cannot be handled, due to over-encryption possibility (as in a non-encrypted form!!)

Detection property

- Any malicious traffic (that is a traffic considered as malicious when not encrypted) must be detected by the MiddleBox

$\underline{\mathsf{Exp}^{det}_{\Delta,\mathcal{A}}(\lambda)}$
$\mathcal{B} \leftarrow \mathsf{RuleGen}(\mathsf{param}, \mathsf{sk}_{\mathsf{SE}}, \mathcal{M});$
$E \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{sk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{R}});$
if $\mathsf{Detect}(\mathsf{param}, \mathsf{sk}_{\mathsf{SP}}, E, \mathcal{B}) = 1$, return 0;
$T \leftarrow \mathsf{Receive}(\mathsf{param}, \mathsf{sk}_{\mathsf{R}}, E);$
if $\mathsf{Detect}(T, \mathcal{M}) = 0$, return 0;
return 1.

Traffic indistinguishability w.r.t. SP (resp. SE)

- It is not feasible for the Service Provider (resp. Security Editor) to learn any information about the traffic, other than it is malicious or safe

$$\underline{\mathsf{Exp}^{sp-tr-ind}_{\Delta,\mathcal{A}}(\lambda)}$$
$b \leftarrow_{\$} \{0,1\}$;
$(T_0, T_1, \mathsf{aux}_{\mathcal{A}}) \leftarrow \mathcal{A}(\mathsf{sk}_{\mathsf{SP}}, \mathsf{param})$; (resp. $\mathcal{A}(\mathsf{sk}_{\mathsf{SE}}, \mathsf{param})$)
if $\mathsf{type}(T_0, T_1) = 0$, return 0;
$E_b \leftarrow \mathsf{Send}(\mathsf{param}, T_b)$;
$b' \leftarrow \mathcal{A}^{\mathsf{Send},\mathsf{RuleGen}}(E_b, \mathsf{aux}_{\mathcal{A}})$;
return $(b = b')$.

Definition (Traffic Type)

Let T_0 and T_1 be two traffics and let $\mathcal{R}$ be a set of rules. We say that T_0 and T_1 are of the *same type*, denoted $\mathsf{type}(T_0, T_1) = 1$, iff $\mathsf{Detect}(\mathsf{param}, T_0, \mathcal{R}) = \mathsf{Detect}(\mathsf{param}, T_1, \mathcal{R})$, including the auxiliary information aux.

Rule indistinguishability w.r.t. SP

- It is not feasible for the Service Provider to learn any information about the rules

$$
\begin{array}{|l|}
\hline
\underline{\mathsf{Exp}^{sp-rul-ind}_{\Delta,\mathcal{A}}(\lambda)} \\
b \leftarrow_{\$} \{0,1\}; \\
(\mathcal{M}_0, \mathcal{M}_1, \mathsf{aux}_{\mathcal{A}}) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{sk}_{\mathsf{SP}}); \\
\mathcal{B}_b \leftarrow \mathsf{RuleGen}(\mathsf{param}, \mathsf{sk}_{\mathsf{SE}}, \mathcal{M}_b); \\
b' \leftarrow \mathcal{A}^{\mathsf{Send}}(\mathcal{B}_b, \mathsf{aux}_{\mathcal{A}}); \\
\text{return } (b = b'). \\
\hline
\end{array}
$$

High-min entropy rule indistinguishability

- It is not feasible for Senders and Receivers to learn any information about the rules

$$\begin{array}{l} \underline{\mathsf{Exp}^{hme-rul-ind}_{\Delta,\mathcal{A}}(\lambda)} \\ b \leftarrow_{\$} \{0,1\}; \\ (\mathcal{M}_0, \mathcal{M}_1) \leftarrow \mathcal{A}_f(\mathsf{param}, \mathsf{sk}_{\mathsf{SP}}, \mathsf{sk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{R}}); \\ \mathcal{B}_b \leftarrow \mathsf{RuleGen}(\mathsf{param}, \mathsf{sk}_{\mathsf{SE}}, \mathcal{M}_b); \\ b' \leftarrow \mathcal{A}_g(\mathcal{B}_b); \\ \text{return } (b = b'). \end{array}$$

Definition (Min-entropy)

A probabilistic adversary $\mathcal{A} = (\mathcal{A}_f, \mathcal{A}_g)$ has *min-entropy* μ if $\forall \lambda \in \mathbb{N}$, $\forall r \in \mathcal{R}$: $\Pr\left[r' \leftarrow \mathcal{A}_f(1^\lambda, b) \ : \ r' = r\right] \ \leq \ 2^{-\mu(\lambda)}$. $\mathcal{A}$ is said to have *high min-entropy* if it has min-entropy μ with $\mu(\lambda) \in \omega(\log \lambda)$.

Problem and first approach

Signature-based detection

- Simple example of an SQL injection
- Example

http://myserver.fr/login?username=seb&password=1234 or (a = a)

- Example of rule

alert tcp any any −> HOMENET PORTHTTP (msg: "SQL Injection Attempt - or a=a"; content: "GET"; httpmethod; uricontent: "or a = a"; nocase; classtype:web-application-attack; sid:3000001; rev:1;)

- The idea is then to search for a specific pattern inside the message
  - simple case: pattern matching
  - complex case: regular expression
- How to proceed if the traffic is encrypted?
  - BlindBox (ACM SIGCOMM 2015): based on MPC, garbled circuits and oblivious transfer $\Longrightarrow$ bad memory complexity, poor time complexity, no rule secrecy

Requirements on encryption

- Server performs encryption and client performs decryption
- MiddleBox performs matching
  - Taking as input an encrypted traffic and a pattern
  - ⇒ We need an encryption scheme with searchable capacity
- But the pattern should not be known to the MiddleBox
  - Due to the rule indistinguishability property
  - ⇒ We need trapdoor-based searchable encryption
  - ⇒ Given T_w and Encrypt(w'), test whether $w = w'$ or not

Decryptable searchable encryption (i)

- Based on a work by Fuhr and Paillier 2007
- F, G, H be three hash functions
- $(q, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_t, e(.,.))$ be a bilinear environment
- Security editor generates $\mathsf{tk} = x' \leftarrow \mathbb{Z}_q$ and publishes $\mathsf{pk}_{\mathsf{SE}} = g_1^{x'}$ and $a \in \mathbb{Z}_q^*$
- Receiver generates $\mathsf{sk}_R = x \leftarrow \mathbb{Z}_q$ and publishes $\widetilde{\mathsf{pk}}_{\mathsf{R}} = g_1^x$
- Key independence between $\mathsf{pk}_{\mathsf{SE}}$ and $\widetilde{\mathsf{pk}}_{\mathsf{R}}$

Decryptable searchable encryption (ii)

- Rule generation: for any word w_i, computes $T_i = F(w_i)^{x'}$
- Traffic encryption: for each token t_i in the traffic, computes

$$\begin{aligned} c_{1,i} &= g_1^{r_i}; \\ (s_1, s_2)_i &= G(\widetilde{\text{pk}}_{\text{R}}^{r_i}); \\ c_{2,i} &= s_{1,i} \oplus t_i; \\ c_{3,i} &= g_1^{s_{2,i}}; \\ u_i &= e(\text{pk}_{\text{SE}}^{s_{2,i}}, F(t_i)); \\ c_{4,i} &= H(u_i) + a \mod q. \end{aligned}$$

- Detection: computes $u_i = e(c_{3,i}, T_j)$ and $a' = c_{4,i} - H(u_i) \mod q$. If $a \neq a'$, then the token is safe.
- Traffic decryption: for each ciphertext, computes

$$\begin{aligned} (s_1, s_2)_i &= G(c_{1,i}^x); \\ t_i &= c_{2,i} \oplus s_{1,i} \end{aligned}$$

Obtained security

- The scheme is detectable provided that there is no collision in the trapdoor generation function
- The scheme is traffic-indistinguishable under the CDH and the GDDHE assumptions in the random oracle model
- The scheme is rule-indistinguishable in the random oracle model

Details about the implementation

- Encrypted pattern matching implies exact pattern matching
  - Sliding window: every character is encrypted multiple times $\Longrightarrow$ better accuracy
  - Delimiter-based: rules and traffic are split according to specified symbols $\Longrightarrow$ more efficient
- Implemented in Java 8, using the Herumi library in C for pairings
- Intel(R) Xeon(R) with a E5-1620 CPU with 4 cores running at 3.70GHz under a 64-bit Linux OS

Obtain performances

- % of detected rules: 75% (only matching)
- Client time: 600 μs for each token
- Server time: 700 μs for each token
- Detection time: 700 μs for each couple (token,rule)

$\Longrightarrow$ 70 s for 3K rules and 1.5KB packet

- Traffic expansion ($|C|/|M|$): 7

First conclusion

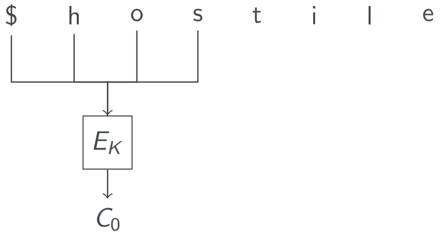
- Publication: "BlindIDS:Market-Compliant and Privacy-Friendly Intrusion Detection System over Encrypted Traffic" by SC, Ada Diop, Nizar Kheir, Marie Paindavoine, Mohamed Sabt at AsiaCCS 2017

- Privacy-friendly $\Longrightarrow$ OK
- Security-aware $\Longrightarrow$ Should be improved
- Market-compliant $\Longrightarrow$ OK
- Practical $\Longrightarrow$ Should be improved

Dealing with security-awareness

How to treat more rules

Solutions based on sliding window method:



keywords

host

hostile

...

- Each C_i can be tested using T_w
- The process must be repeated for each possible length of keywords

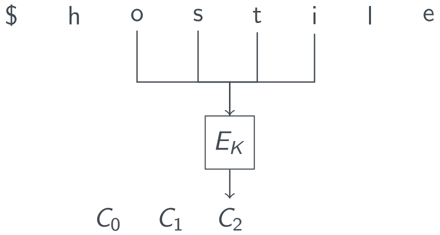
How to treat more rules

Solutions based on sliding window method:

- Each C_i can be tested using T_w
- The process must be repeated for each possible length of keywords

How to treat more rules

Solutions based on sliding window method:



keywords

host

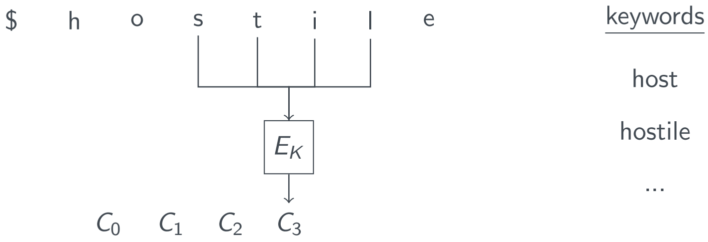
hostile

...

- Each C_i can be tested using T_w
- The process must be repeated for each possible length of keywords

How to treat more rules

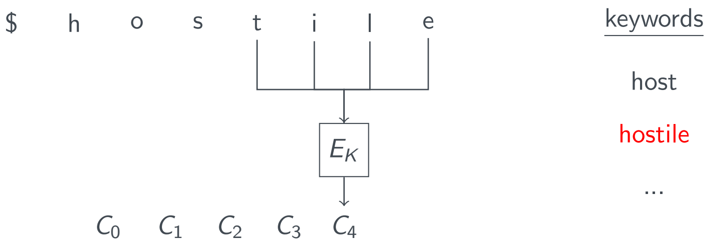
Solutions based on sliding window method:



- Each C_i can be tested using T_w
- The process must be repeated for each possible length of keywords

How to treat more rules

Solutions based on sliding window method:



- Each C_i can be tested using T_w
- The process must be repeated for each possible length of keywords

How to do better

- Anonymous Predicate Encryption enables to encrypt for a set of attributes $A_1,\dots,A_n$
- A secret key sk_P is associated with a predicate P:

$$C \text{ can be decrypted} \Leftrightarrow P(A_1,\dots,A_n) = 1$$

- Efficient solutions exist for predicate P such that:

$$P(A_1,\dots,A_n) = 1 \Leftrightarrow A_i = Y_i,\ \forall i \in \mathcal{I} \subset [1,n]$$

Dealing with Data Streams

Each character is considered as an attribute

plaintext	$	h	o	s	t	i	l	e
$P_{\texttt{host},0}$	h	o	s	t	*	*	*	*
$P_{\texttt{host},1}$	*	h	o	s	t	*	*	*
$P_{\texttt{host},2}$	*	*	h	o	s	t	*	*
$P_{\texttt{host},3}$	*	*	*	h	o	s	t	*
$P_{\texttt{host},4}$	*	*	*	*	h	o	s	t

keyword: host

- A predicate is defined for each keyword and each possible offset
- $\mathsf{sk}_{P_{\texttt{host},j}}$ enables to check if the plaintext contains `host` at offset j
- Secret keys must be issued for each possible offset
- Not so good...

SEST

Introduction of a new primitive, Searchable Encryption with Shiftable Trapdoors

- Similar to predicate encryption
- A Test algorithm run on $E_K(b_1 \dots b_m)$ and a trapdoor for $W = w_1 \dots w_\ell$ returns

$$\mathcal{J} = \{j : b_{j+1} \dots b_{j+\ell} = w_1 \dots w_\ell\}$$

- Security requires indistinguishability of two encrypted bitstrings, unless issued trapdoors enable trivial distinctions
- Proposed construction based on bilinear pairing, proven secure in the generic group model

Obtain performances

- % of detected rules: 90% (full (but only) matching)
- Client time: 12.5 μs for each byte
- Server time: 25 μs for each byte
- Detection time: 750 μs for each possible position

$\Longrightarrow$ 844 s for 3K rules and 1.5KB packet

- Traffic expansion ($|C|/|M|$): 64

Next conclusion

- Publication: "Pattern Matching on Encrypted Streams" by Nicolas Desmoulins, Pierre-Alain Fouque, Cristina Onete, Olivier Sanders at Asiacrypt 2018

- Privacy-friendly $\Longrightarrow$ OK
- Security-aware $\Longrightarrow$ OK
- Market-compliant $\Longrightarrow$ OK
- Practical $\Longrightarrow$ Should be improved

Treating regular expressions...

- Using (interactive) functional encryption
  - Setup(1^λ): master secret key msk and master public key mpk
  - IKeyGen($\mathcal{AUT}$(msk),$\mathcal{U}$(mpk, f)): interactive protocol to obtain functional key sk_f (New!!)
  - Enc(mpk, m): ciphertext c.
  - Dec(mpk, sk_f, c): z such that $z = f(m)$
- In practice
  - Sender encrypts and Receiver has full decryption
  - Security editor is the authority managing the master secret key msk
  - Service provider executes functional decryption Dec

Functionalities and security

- What about functionalities?
  - Inner product permits to test equal patterns
  - More general functions permit to treat regular expression
- What about security properties?
  - Message privacy for traffic indistinguishability
  - Blindness for rule indistinguishability (New!!)

Next conclusion

- In submission: "Blind Functional Encryption" by SC, Adel Hamdi, Fabien Laguillaumie
- Construction for inner product
  - Privacy-friendly $\Longrightarrow$ OK
  - Security-aware $\Longrightarrow$ Should be improved
  - Market-compliant $\Longrightarrow$ OK
  - Practical $\Longrightarrow$ Should be improved
- Generic construction (from FHE and ZKPK)
  - Privacy-friendly $\Longrightarrow$ OK
  - Security-aware $\Longrightarrow$ OK (100%!)
  - Market-compliant $\Longrightarrow$ OK
  - Practical $\Longrightarrow$ Should strongly be improved!!!

Dealing with practicality

Using symmetric encryption techniques

- A traffic T is divided into tokens t_j
- A secret key s is shared between SE, S and R
  - Used by SE to compute a blinded rules B_i for each searchable pattern
  - Used by S to compute a blinded version p_j of each token t_j
  - Detection becomes a simple match, using a deterministic algorithm
  - Using a pseudorandom permutation (PRP) F

action(actor)	**inputs**	**actions**
RuleGen(SE)	rules r_i, key s	$B_i = F(\mathsf{s}, r_i)$
Send(S)	tokens t_j, key s	$p_j = F(\mathsf{s}, t_j)$
Receive(R)	traffic p_j, key s	$t_j = F^{-1}(\mathsf{s}, p_j)$
Detect(SP)	rules B_i, traffic p_j	$B_i = p_j$?

Managing traffic indistinguishability w.r.t. SE

- Encapsulation technique avoids breaking traffic indistinguishability by SE
- Using of a pseudorandom function (PRF) G in counter mode for each p_j
- SP should obtain both B_i and p_j for detection
- Key K shared by SP, S and R

action(actor)	**inputs**	**actions**
RuleGen(SE)	rules r_i, key s	$B_i = F(\mathsf{s}, r_i)$
Send(S)	tokens t_j, keys (s, K)	$p_j = F(\mathsf{s}, t_j)$ $q_j = G(\mathsf{K}, j) \oplus p_j$
Receive(R)	traffic q_j, keys (s, K)	$p_j = q_j \oplus G(\mathsf{K}, j)$ $t_j = F^{-1}(\mathsf{s}, p_j)$
Detect(SP)	rules B_i, key K traffic q_j	$p_j = G(\mathsf{K}, j) \oplus q_j$ $B_i = p_j$?

Managing randomness

- Addition of a random counter $c \leq C$ used with the PRP F
- SE generates C blinded tokens for each rule r_i during RuleGen
- Addition of a true random salt to G

action(actor)	**inputs**	**actions**
RuleGen(SE)	rules r_i, key s	$B_{i,k} = F(\text{s}, r_i \Vert c_k)$
Send(S)	tokens t_j, keys (s, K), counter c, random salt	$p_j = F(\text{s}, t_j \Vert c)$ $q_j = G(\text{K}, \text{salt} + j) \oplus p_j$
Receive(R)	traffic q_j, keys (s, K), counter c, salt	$p_j = q_j \oplus G(\text{K}, \text{salt} + j)$ $t_j \Vert c = F^{-1}(\text{s}, p_j)$
Detect(SP)	rules B_i, key K, traffic q_j, salt	$p_j = G(\text{K}, \text{salt}+j) \oplus q_j$ $B_i = p_j$?

Non-inversibility of blinded rules

- A fraudulent sender or receiver can inverse the $B_{i,k}$'s $\Longrightarrow$ replace F by a non-reversible pseudorandom function
- Receiver is no more able to decrypt the traffic $\Longrightarrow$ Additional TLS encryption with a shared k
- Sender can send two different traffics $\Longrightarrow$ SP hashes the encrypted tokens p_j

action(actor)	inputs	actions
RuleGen(SE)	rules r_i, key s	$B_{i,k} = F(\mathsf{s}, r_i \| c_k)$
Send(S)	tokens t_j, keys $(\mathsf{s}, \mathsf{K}, \mathsf{k})$, counter c, random salt	$e = \mathsf{TLS}(\mathsf{k}, \{t_j\}_j)$, $p_j = F(\mathsf{s}, t_j \| c)$, $q_j = G(\mathsf{K}, \mathsf{salt} + j) \oplus p_j$
Receive(R)	traffic e, keys $(\mathsf{s}, \mathsf{K}, \mathsf{k})$, counter c, salt, hash h_j	$\{t_j\}_j = \mathsf{TLS}^{-1}(\mathsf{k}, e)$, $p_j = F(\mathsf{s}, t_j \| c)$, $h_j = \mathcal{H}(p_j)$?
Detect(SP)	rules B_i, key K, traffic q_j, salt	$p_j = G(\mathsf{K}, \mathsf{salt} + j) \oplus q_j$, $B_i = p_j$?, $h_j = \mathcal{H}(p_j)$

Decreasing the number of blinded rules

- SE has to compute a number of encrypted rules proportional to the number of couples S/R times the constant $C \Longrightarrow$ make use of a broadcast encryption scheme $\mathtt{BE}$

action(actor)	inputs	actions
RuleGen(SE)	rules r_i, master key mk	$(\mathsf{s}, \mathsf{Hdr}) = \mathtt{BE}.\mathsf{Enc}(\mathsf{mk}, I)$ $B_{i,k} = F(\mathsf{s}, r_i \| c_k)$
Send(S)	tokens t_j, keys $(\mathsf{sk}_n, \mathsf{K}, \mathsf{k})$, counter c, random salt	$\mathsf{s} = \mathtt{BE}.\mathsf{Dec}(\mathsf{sk}_n, \mathsf{Hdr})$ $e = \mathsf{TLS}(\mathsf{k}, \{t_j\}_j)$, $p_j = F(\mathsf{s}, t_j \| c)$, $q_j = G(\mathsf{K}, \mathsf{salt} + j) \oplus p_j$
Receive(R)	traffic e, keys $(\mathsf{sk}_{\tilde{n}}, \mathsf{K}, \mathsf{k})$, counter c, salt, hash h_j	$\mathsf{s} = \mathtt{BE}.\mathsf{Dec}(\mathsf{sk}_{\tilde{n}}, \mathsf{Hdr})$, $\{t_j\}_j = \mathsf{TLS}^{-1}(\mathsf{k}, e)$, $p_j = F(\mathsf{s}, t_j \| c)$, $h_j = \mathcal{H}(p_j)$?
Detect(SP)	rules B_i, key K, traffic q_j, salt	$p_j = G(\mathsf{K}, \mathsf{salt} + j) \oplus q_j$, $B_i = p_j$?, $h_j = \mathcal{H}(p_j)$

Obtained security

- The scheme is detectable if hash function is collision resistant
- The scheme is traffic-indistinguishable if broadcast encryption is indistinguishable, F and G are pseudorandom
- The scheme is rule-indistinguishable if broadcast encryption is indistinguishable, F is pseudorandom and fixed-key PRF F is one-way

Obtain performances

- % of detected rules: 75% (only matching)
- Client time: 200 *ns* for each token
- Server time: 250 *ns* for each token
- Detection time: 10 *ns* for each couple (token,rule)

$\Longrightarrow$ 1.5 μs for 3K rules and 1.5KB packet (compare to 70 *s* and 844 *s*!)

- Traffic expansion ($|C|/|M|$): 2

Next conclusion (again)

- In submission: "Towards Truly Practical Intrusion Detection System over Encrypted Traffic" by SC, Chaoyun Li

- Privacy-friendly $\Longrightarrow$ OK
- Security-aware $\Longrightarrow$ Should be improved
- Market-compliant $\Longrightarrow$ OK
- Practical $\Longrightarrow$ OK
- But take care of key management and counter...

Final conclusion: ANR PRESTO project

- Consortium: ENS (leader), IMT, LORIA, Orange, 6cure
- Duration: 4 years
- Main techniques: searchable encryption, functional encryption, (homomorphic encryption)
- Use cases (on-line and off-line): denial of service attacks, content filtering, forensic analysis
- Standardization...

Thank you

Thanks to Nicolas Desmoulins (Orange), Aïda Diop (Orange, Telecom SudParis), Pierre-Alain Fouque (Université de Rennes), Adel Hamdi (Orange, ENS Lyon) Nizar Kheir (Thales), Fabien Laguillaumie (ENS Lyon), Chaoyun Li (KUL), Cristina Onete (Université de Limoges), Marie Paindavoine (Thales), Mohamed Sabt (IRISA), Olivier Sanders (Orange)