Pattern Matching on Encrypted Streams

Nicolas Desmoulins
Orange Labs

Pierre-Alain Fouque
Université de Rennes

Cristina Onete
Université de Limoges

Olivier Sanders
Orange Labs

Asiacrypt 2018, December 03, 2018

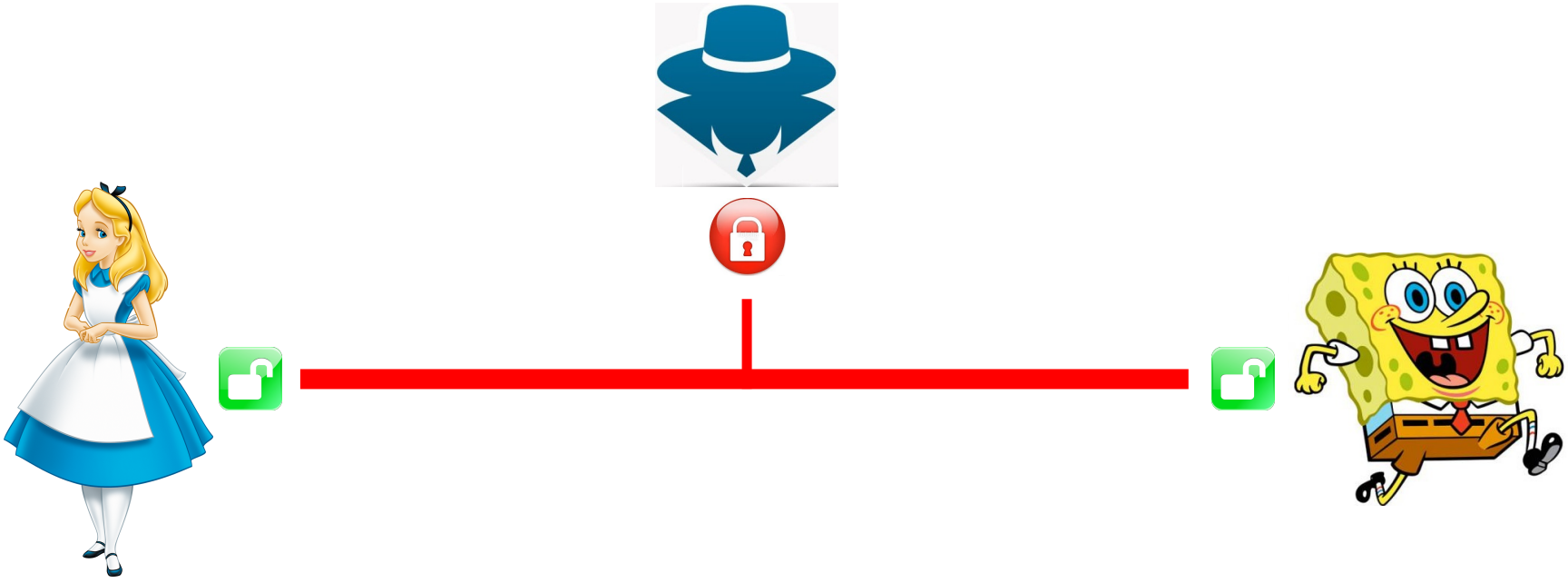
orange™

Agenda

- Context
- Our Contribution
- Conclusion

Context

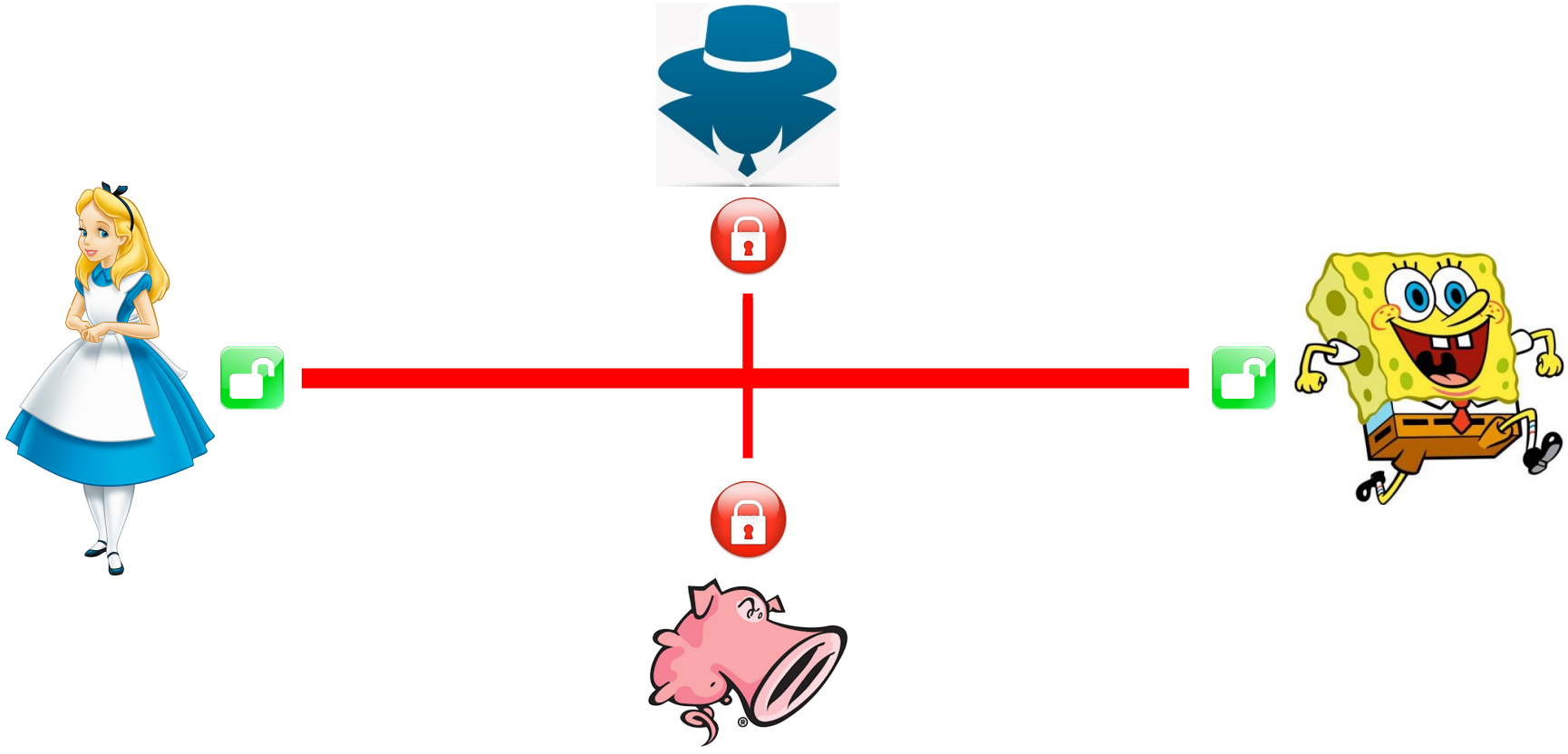
End-to-End Encryption



More and more encrypted data

- 50% of worldwide traffic is encrypted, 80% expected by 2020
- development of encrypted messaging services (WhatsApp, Signal,...)

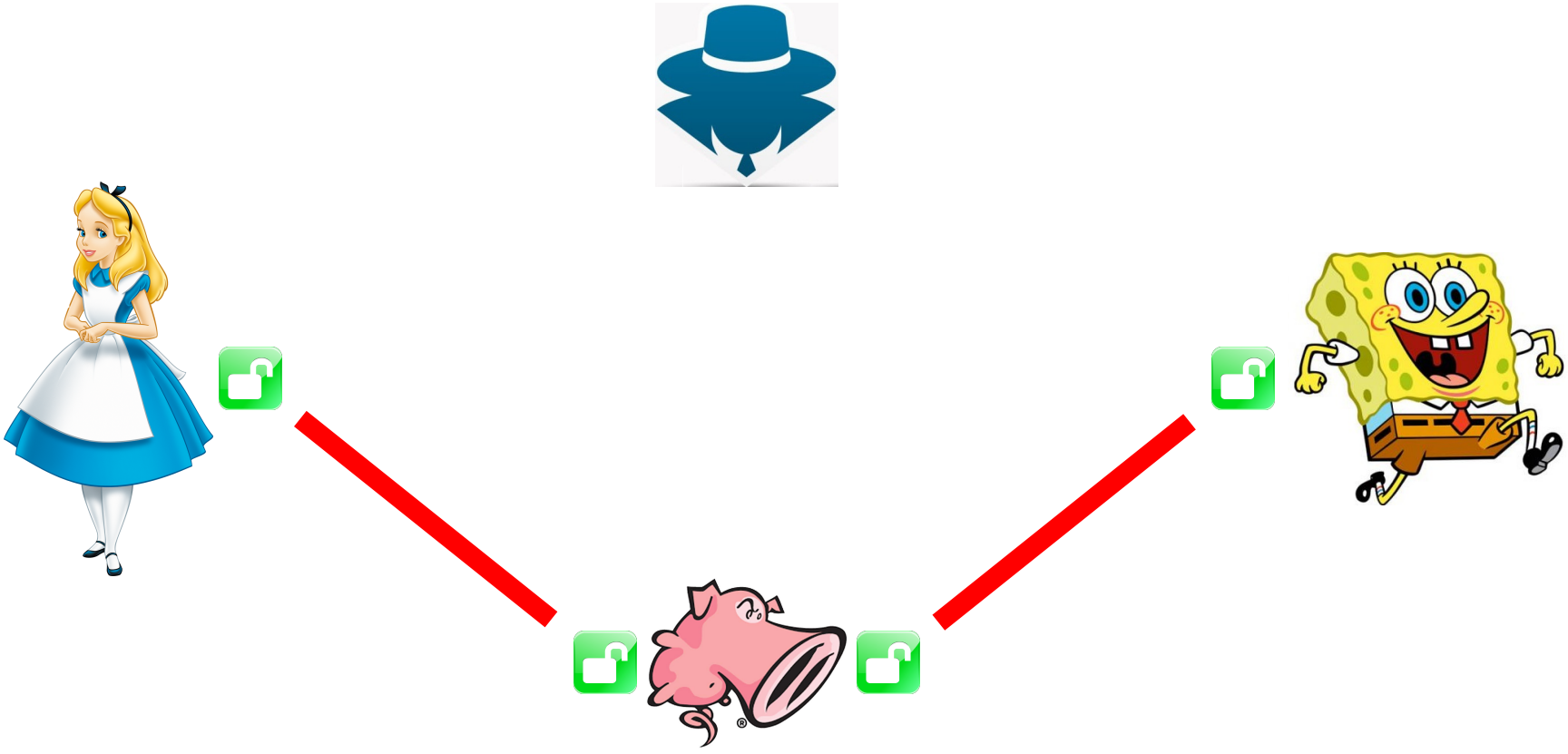
End-to-End Encryption



Standard encryption protocols designed to prevent any processing

- no possible tradeoff between privacy and functionalities
- incompatible with security applications such as IDS

End-to-End Encryption



Current solutions imply decryption by a gateway

- the gateway can access all data exchanged through the channel
- what is the point of end-to-end encryption?

Generic Solutions

- Processing of encrypted data has been extensively studied
- Generic solutions exist but are very complex
  - fully homomorphic encryption: high computational cost
  - multi-party computation: requires interaction with the gateway/ high communication cost
- Solutions tailored to specific tasks can significantly improve efficiency

Pattern Matching

- Example of Snort rules:
  - alert tcp (msg:"MALWARE-BACKDOOR - Dagger_ 1.4.0"; content:"2| 00 00 00 06 00 00 00 | Drives | 24 00 |",depth 16;)
  - alert tcp (msg:"MALWARE-BACKDOOR QAZ Worm Client Login access"; content:"qazwsx.hsq";)
- Pattern matching is essential to several other applications
  - searches on genomic data
  - filtering content
  - ...
- Solutions exist but they are unsuitable for data streams

Searchable Encryption

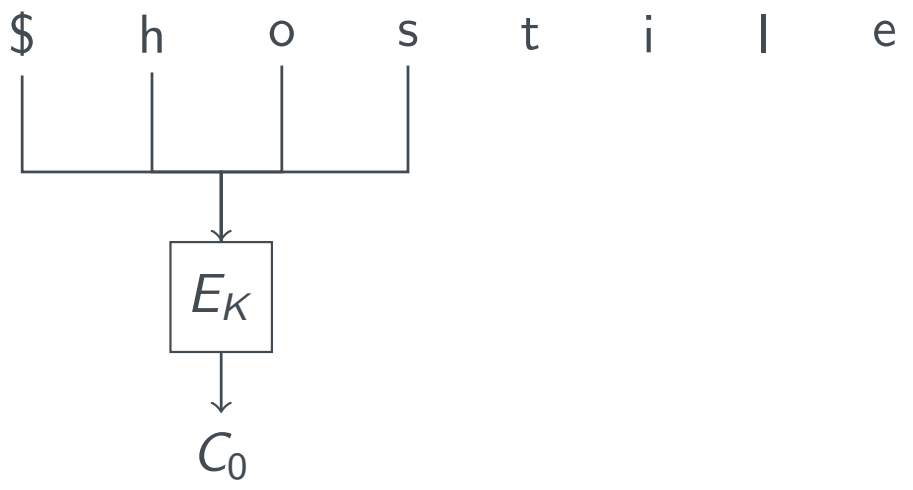
- Symmetric searchable encryption for database is very efficient but:
  - documents must be associated with keywords

    $\Rightarrow$ how to select relevant keywords in our context?
  - not designed for on the fly encryption

    $\Rightarrow$ unsuitable for data streams
- Standard searchable encryption enables to issue tokens td_W for specific keywords W
  - given td_w, the gateway can check whether $C = E_K(W)$
  - the gateway learns no information beyond the result of this query

Dealing with Data Streams

Current solutions follow the sliding window method:



keywords

host

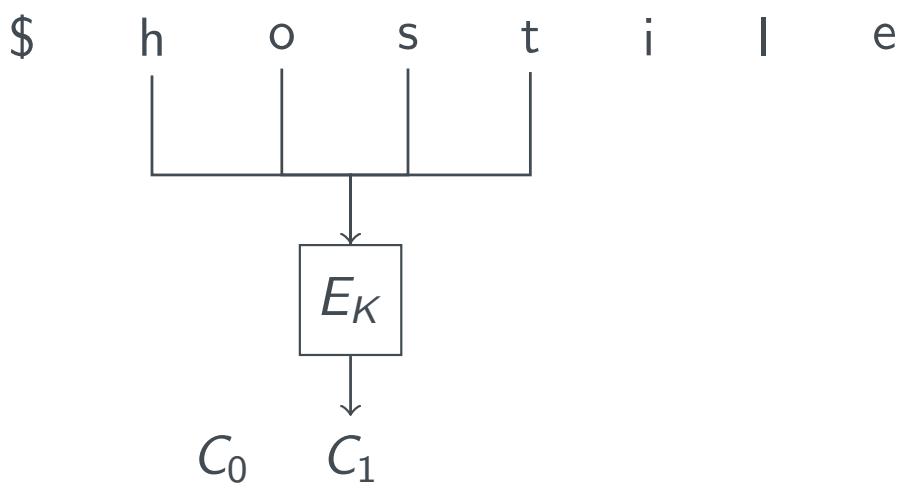
hostile

...

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

Dealing with Data Streams

Current solutions follow the sliding window method:



keywords

host

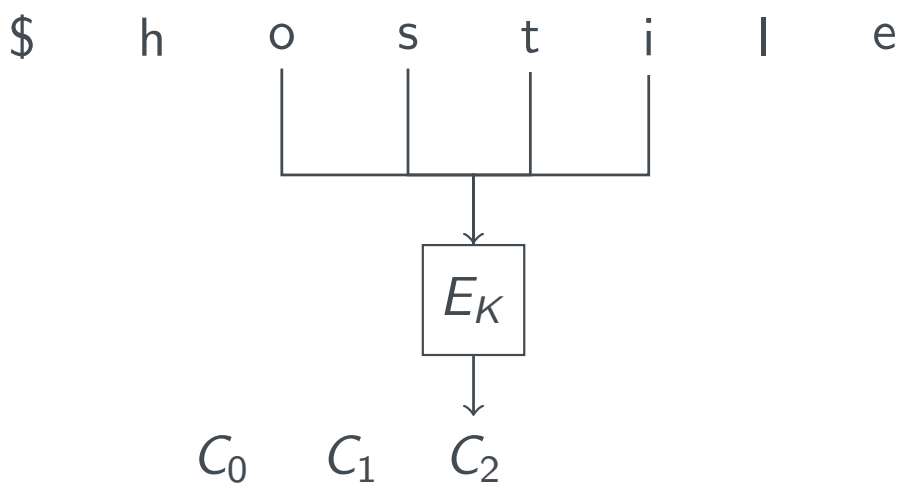
hostile

...

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

Dealing with Data Streams

Current solutions follow the sliding window method:



keywords

host

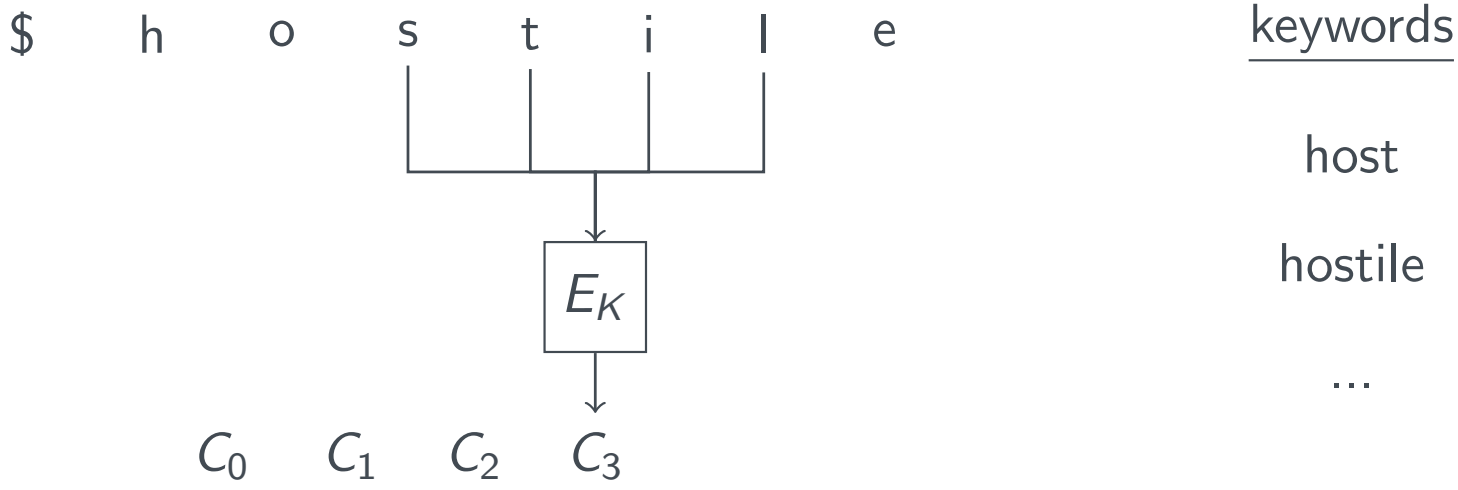
hostile

...

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

Dealing with Data Streams

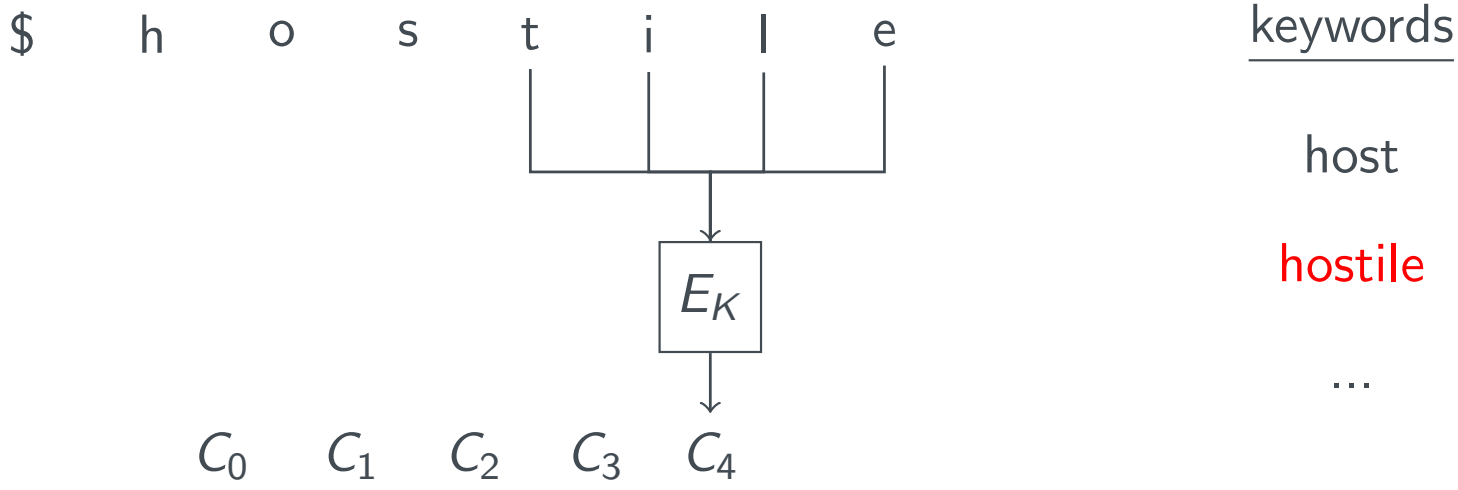
Current solutions follow the sliding window method:



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

Dealing with Data Streams

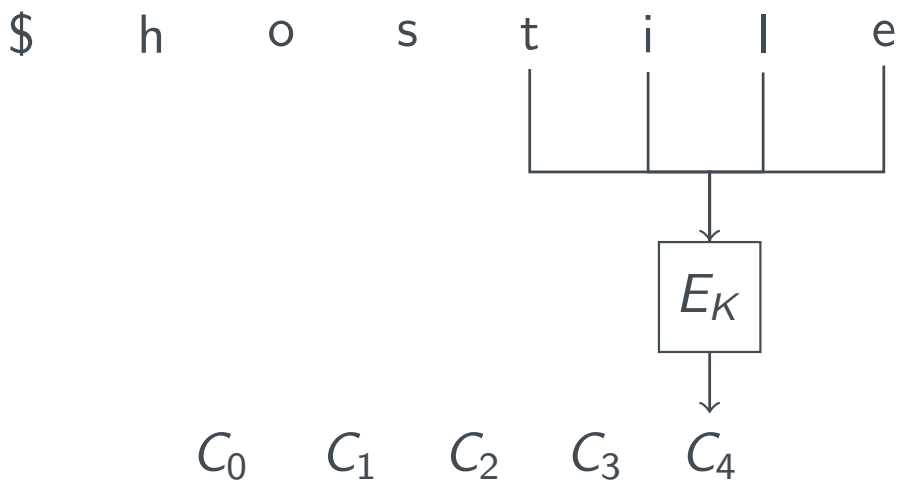
Current solutions follow the sliding window method:

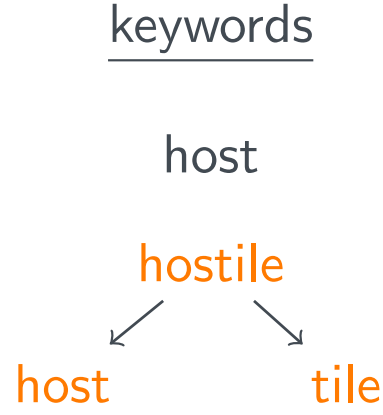


- Each C_i can be tested using td_W
- The process must be repeated for each possible length of keywords

Dealing with Data Streams

Current solutions follow the sliding window method:





- Each C_i can be tested using td_W
- Splitting keywords in smaller ones of fixed length severely harms privacy: thoussands of (potentially long) keywords to split

Anonymous Predicate Encryption

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

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

- No other information is leaked on the attributes of C
- Efficient solutions exist for predicate P such that:

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

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
- $\mathrm{sk}_{P_{\texttt{host},j}}$ enables to check if the plaintext contains `host` at offset j
- Secret keys must be issued for each possible offset

Our Goals

We want an encryption scheme such that:

- pattern matching is possible anywhere in the ciphertext
- Encryption is independent of the searchable keywords

  $\Rightarrow$ ciphertexts should be compatible with keywords of any length

- td_W allows for searches at any possible offset

  $\Rightarrow$ not 1 token by possible offset

Our Contribution

SEST

We introduce 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.

Bilinear Groups

We construct a scheme based on asymetric bilinear groups

- Bilinear Groups: 3 groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of prime order p along with a map e such that

$$\forall (g, \widetilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2 \text{ and } a, b \in \mathbb{Z}_p \; e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{a \cdot b}$$

$$e(g, \widetilde{g}) = 1_{\mathbb{G}_T} \Longrightarrow g = 1_{\mathbb{G}_1} \text{ or } \widetilde{g} = 1_{\mathbb{G}_2}$$

- Asymmetry : no efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ exists
- Such groups are easily instantiated using elliptic curves

Intuition - Step 1

Let us consider bitstrings $B = b_1 \dots b_n$

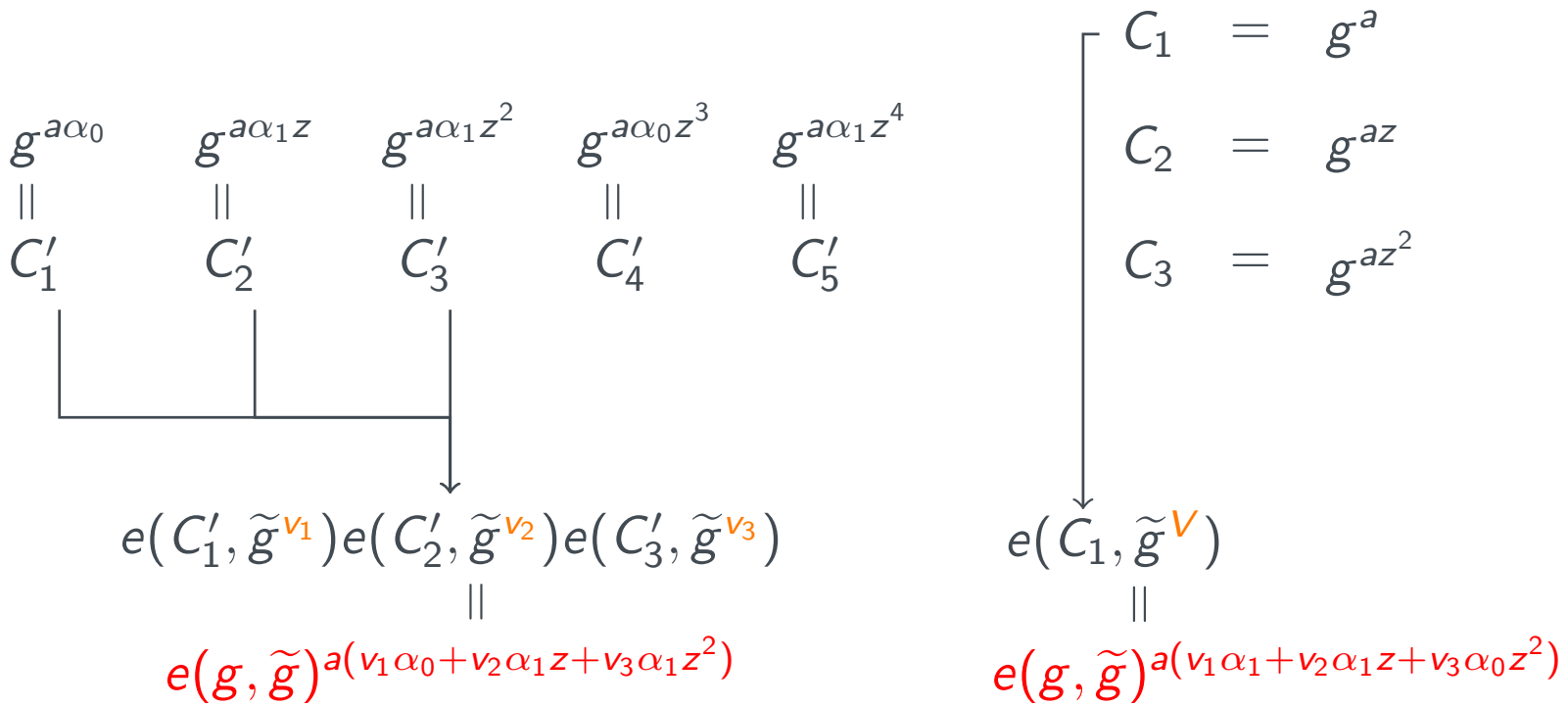
- We define secret encodings $\alpha_0, \alpha_1 \in \mathbb{Z}_p$ associated with 0 and 1
- We select a secret $z \in \mathbb{Z}_p$ defining a public "basis" $(g, g^z, \dots, g^{z^{n-1}})$ of $\mathbb{G}_1^n$
- Encryption of B is performed by
  1. randomizing the basis
     $$(C_1, \dots C_n) \leftarrow (g^a, (g^z)^a, \dots, (g^{z^n})^a) \text{ for } a \xleftarrow{\$} \mathbb{Z}_p$$
  2. "projecting" B on this basis
     $$(C'_1, \dots C'_n) \leftarrow ([(g^z)^a]^{\alpha_{b_1}}, \dots, [(g^{z^n})^a]^{\alpha_{b_n}})$$

Trapdoors

- The secret key sk is $\{z, \alpha_0, \alpha_1\}$
- To issue a trapdoor td_W for $W = w_1 \ldots w_\ell$
  1. generate random scalars $v_1, \ldots, v_\ell \xleftarrow{\$} \mathbb{Z}_p$
  2. compute $\widetilde{g}^{v_i}$ in $\mathbb{G}_2$
  3. compute $\widetilde{g}^{V}$ with $V = \sum_{i=1}^{\ell} v_i \cdot \alpha_{w_i} \cdot z^{i-1}$
  4. return $\mathrm{td}_W = \{\widetilde{g}^{v_1}, \ldots, \widetilde{g}^{v_\ell}, \widetilde{g}^{V}\}$
- Each trapdoor is associated with a random polynomial V
- Random coefficients v_i are used to prevent trapdoor forgeries

Intuition - Step 2

B	0	1	1	0	1
W	1	1	0		



Consecutive C'_i can be aggregated to generate $e(g, \widetilde{g})^{aP(z,\alpha_0,\alpha_1)}$

Intuition - Step 2

B	0	1	1	0	1
W		1	1	0	

$$C_1 = g^{a}$$

$$C_2 = g^{az}$$

$$C_3 = g^{az^2}$$

$$g^{a\alpha_0} \quad g^{a\alpha_1 z} \quad g^{a\alpha_1 z^2} \quad g^{a\alpha_0 z^3} \quad g^{a\alpha_1 z^4}$$

$$\| \quad \| \quad \| \quad \| \quad \|$$

$$C_1' \quad C_2' \quad C_3' \quad C_4' \quad C_5'$$

$$e(C_2', \tilde{g}^{v_1})\, e(C_3', \tilde{g}^{v_2})\, e(C_4', \tilde{g}^{v_3}) = e(g, \tilde{g})^{a(v_1\alpha_1 z + v_2\alpha_1 z^2 + v_3\alpha_0 z^3)}$$

$$e(C_2, \tilde{g}^{V}) = e(g, \tilde{g})^{a(v_1\alpha_1 z + v_2\alpha_1 z^2 + v_3\alpha_0 z^3)}$$

B contains $W \Leftrightarrow P = z^* V$

Intuition - Step 2

B	0	1	1	0	1
W			1	1	0

$$C_1 = g^{a}$$

$$C_2 = g^{az}$$

$$C_3 = g^{az^2}$$

$$g^{a\alpha_0} \quad g^{a\alpha_1 z} \quad g^{a\alpha_1 z^2} \quad g^{a\alpha_0 z^3} \quad g^{a\alpha_1 z^4}$$

$$\| \quad \| \quad \| \quad \| \quad \|$$

$$C_1' \quad C_2' \quad C_3' \quad C_4' \quad C_5'$$

$$e(C_3', \widetilde{g}^{v_1})\, e(C_4', \widetilde{g}^{v_2})\, e(C_5', \widetilde{g}^{v_3}) = e(g, \widetilde{g})^{a(v_1\alpha_1 z^2 + v_2\alpha_0 z^3 + v_3\alpha_1 z^4)}$$

$$e(C_3, \widetilde{g}^{V}) = e(g, \widetilde{g})^{a(v_1\alpha_1 z^2 + v_2\alpha_1 z^3 + v_3\alpha_0 z^4)}$$

Keywords of any size can be tested

Features

- Our construction can handle any kind of strings (bytestrings, ...)
- Our construction supports wildcards
  - Let $W = w_1 \ldots w_{k-1} * w_{k+1} \ldots w_\ell$
  - The associated coefficient v_k in td_W is set to 0

    $k - th$ element of a substring no longer taken into account
- We can handle certain types of regular expressions by defining special encodings

  $$e.g.\ \epsilon([0-9]) = \beta_1 \in \mathbb{Z}_p,\ \epsilon([a-z]) = \beta_2,\ \ldots$$
- Our construction is proven secure in the generic group model

Optimization

Let $B = b_1 \ldots b_n$ and $W = w_1 \ldots w_\ell$

- Previous detection procedure requires $(n - \ell + 1)(\ell + 1)$ pairings:

$$\forall j : e(C'_{j+1}, \widetilde{g}^{v_1}) \ldots e(C'_{j+\ell}, \widetilde{g}^{v_\ell}) = e(C_{j+1}, \widetilde{g}^{V})$$

- Security requires that V has distinct coefficients

$$V = v_1 \alpha_{w_1} + v_2 \alpha_{w_2} z \ldots + v_\ell \alpha_{w_\ell} z^{\ell-1}$$

- If $w_i \neq w_k$, we can set $v_i = v_k$ and merge the corresponding pairings:

$$e(C'_{j+i}, \widetilde{g}^{v_i}) \cdot e(C'_{j+k}, \widetilde{g}^{v_k}) = e(C'_{j+i} \cdot C'_{j+k}, \widetilde{g}^{v_i})$$

- Divides the number of pairings by up to 256 for bytestrings.

Further Optimizations

- Several pairings involved in the same equation
  - the final exponentiations can be merged
  - some steps of the Miller loop can be merged
- Most computations are embarrassingly parallelizable
- The coefficients v_i can be selected so that $V = 0$
  - Divides by 2 the size of the ciphertext
  - Forbids constant pattern (*e.g.* *aaaaaaaaaa*)

Conclusion

Conclusion

- We propose a new primitive, tailored to pattern matching on encrypted data
- We propose a construction based on bilinear groups
- We avoid the inherent problems of the sliding window method and of offsets for predicate
- We prove the security of our scheme in the generic group model