FSVPDsee: A Forward Secure Publicly Verifiable Dynamic SSE Scheme

Laltu Sardar[1] Sushmita Ruj [1,2]

[1]Indian Statistical Institute, Kolkata, India

[2]CSIRO Data61, Australia

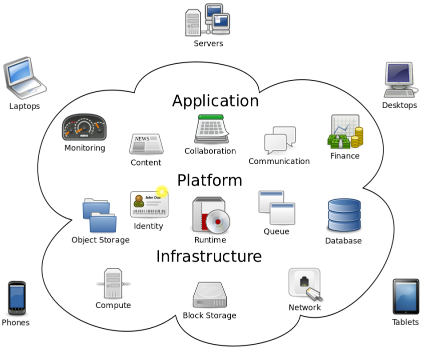
ProvSec 2019, Cairns, Australia

Dependency on Cloud Computing and Storage Services

Azure

aws

Google Drive



Cloud computing

IBM Cloud

Google Cloud

Dropbox

Can we TRUST Cloud Service Providers?

TRUST?

Can we TRUST Cloud Service Providers?

TRUST?

Read Data

Sell Data

Hacked

Manipulation

Can we TRUST Cloud Service Providers?

TRUST?

Read Data

Sell Data

Hacked

Manipulation

How to be SAFE?

How to be SAFE?

Encrypt data

Send Encrypted Data

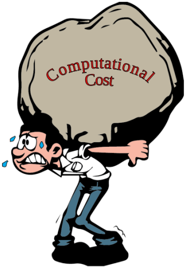
Store in cloud

How to be SAFE?

Encrypt data

Send Encrypted Data

Store in cloud



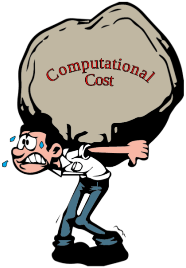
Cloud can't Search!

How to be SAFE?

Encrypt data

Send Encrypted Data

Store in cloud



Cloud can't Search!

Solution: Searchable Encryption

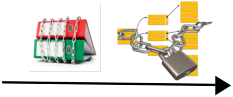
Searchable Encryption Idea

Initially

Build Search Index

1. Encrypt data and Index

2. Send



Encrypted Data & Search Index

3. Store data and index

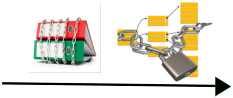
Searchable Encryption Idea

Initially

Build Search Index

1. Encrypt data and Index

2. Send



Encrypted Data & Search Index

3. Store data and index

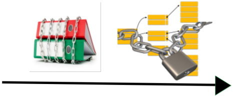
Searchable Encryption Idea

Initially

Build Search Index

1. Encrypt data and Index

2. Send



Encrypted Data & Search Index

3. Store data and index

For Search:

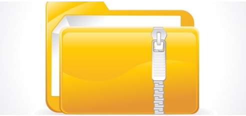
1. Client Sends

Search Token

2. Cloud Search

over Encrypted Index

3.Client received



Result

Presence of Malicious adversary

Previous works

- Plenty of works on static SSE and dynamic SSE
- Dynamic SSE– Supports updates on database, popular
- Assumes– **Honest-but-curious** Server which Follows the protocol but wants to Learn data

Presence of Malicious adversary

Previous works

- Plenty of works on static SSE and dynamic SSE
- Dynamic SSE– Supports updates on database, popular
- Assumes– **Honest-but-curious** Server which Follows the protocol but wants to Learn data

What happen if the server is malicious?

- Computation over data requires cost
- Can not provide free service as it can not sell data
- Can skip search and return wrong result, incomplete result

Presence of Malicious adversary

Previous works

- Plenty of works on static SSE and dynamic SSE
- Dynamic SSE– Supports updates on database, popular
- Assumes– **Honest-but-curious** Server which Follows the protocol but wants to Learn data

What happen if the server is malicious?

- Computation over data requires cost
- Can not provide free service as it can not sell data
- Can skip search and return wrong result, incomplete result

!!! Verification Needed !!!

Can Public also verify?

Private verifiability

- Only the owner/querier can verify
- Verifier has to do most of the computation

Public verifiability

- Any of the owner, querier or third party **Auditor** can verify.
- Most of the Computation for verifiability can be Outsourced
- Result not revealed

Forward Privacy & Client storage

Forward Privacy

- adding a keyword-document pair does not reveal any information about the previous search result with that keyword.

Attacks– when no Forward Privacy

- File-injection attack [ZKP16]: Cloud Can inject files
- Can recover queries

Forward Privacy & Client storage

Forward Privacy

- adding a keyword-document pair does not reveal any information about the previous search result with that keyword.

Attacks– when no Forward Privacy

- File-injection attack [ZKP16]: Cloud Can inject files
- Can recover queries

!!! Forward Privacy Needed !!!

Forward Privacy & Client storage

Forward Privacy

- adding a keyword-document pair does not reveal any information about the previous search result with that keyword.

Attacks– when no Forward Privacy

- File-injection attack [ZKP16]: Cloud Can inject files
- Can recover queries

!!! Forward Privacy Needed !!!

Need of Owner Storage Reduction

- Owner should able to search and verify from lightweight devices
- Should require storage and computation as less as possible

Forward Privacy & Client storage

Forward Privacy

- adding a keyword-document pair does not reveal any information about the previous search result with that keyword.

Attacks– when no Forward Privacy

- File-injection attack [ZKP16]: Cloud Can inject files
- Can recover queries

!!! Forward Privacy Needed !!!

Need of Owner Storage Reduction

- Owner should able to search and verify from lightweight devices
- Should require storage and computation as less as possible

Challenge– Enabling Public verifiability without extra client storage !

Our Contribution

Verifiable SSE

- Described problem with existing static SSE schemes
- Proposed a generic efficient solution
- Solution that no needs of any extra storage at owner side

Forward Secure Verifiable DSSE

- Proposed a **GENERIC** publicly verifiable dynamic SSE scheme (Ψ_f)
- very efficient and easy to integrate

Extra Benefits

- **The owner does not use any extra storage** than the embedded schemes.
- very effective and efficient for a resource constrained client.

Verifiable SSE and DSSE schemes

Table: Different verifiable SSE and DSSE schemes

Data Type	static				dynamic			
Query Type	single		complex		single		complex	
Verification	private	public	private	public	private	public	private	public
Schemes	[CG12], [CYG+15], [OK17], [LLL+18],	[SK19]	[WCS+18], [LZQ+18], [XZX18]	[SK19]	[YK17], [BFP16]	[MWWM19], Ψ_f	[ZLW16]	[JZGL15]
Forward Private	not applicable				[YK17], Ψ_f			

- Our scheme is Publicly verifiable single keyword search scheme.
- The only forward private scheme [YK17] is privately verifiable

Used Cryptographic tools

Bilinear Map

Let $\mathbb{G} =< g >$ and $\mathbb{G}_T$ be two **cyclic** groups of prime order q.
$\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an *admissible non-degenerate bilinear map* if–

- $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$, $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$ (bilinearity)
- $\hat{e}(g, g) \neq 1$ (non-degeneracy)
- $\hat{e}$ can be computed efficiently.

Used Cryptographic tools

Bilinear signature

Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map where $|\mathbb{G}| = |\mathbb{G}_T| = q$, a prime and $\mathbb{G} =< g >$. A bilinear signature (BLS) scheme $\mathcal{S}$=(**Gen**, **Sign**, **Verify**) is a tuple of three algorithms as follows.

- $(sk, pk) \leftarrow$ **Gen**: It selects $\alpha \stackrel{\$}{\leftarrow} [0, q-1]$. It keeps the $\boxed{\text{private key } sk = \alpha}$, publishes the $\boxed{\text{public key } pk = g^{\alpha}}$.
- $\sigma \leftarrow$ **Sign**(sk, m): Given $sk = \alpha$, and some message m, it outputs the $\boxed{\text{signature } \sigma = (\mathcal{H}(m))^{\alpha} = (g^m)^{\alpha}}$ where $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}$ is a bilinear hash defined by $\mathcal{H}(m) = g^m$.
- $\{0/1\} \leftarrow$ **Verify**(pk, m, σ): Return whether $\boxed{\hat{e}(\sigma, g) = \hat{e}(\mathcal{H}(m), g^{\alpha})}$

Verifiablity on Static SSE Schemes

Traditional Verifiable Schemes with client storage

- Target– Given w, verify $R_w = \{D_1^w, D_2^w, \ldots, D_n^w\}$
- For each w, stores $\boxed{digest(w) \leftarrow H(D_1^w || D_2^w || \ldots || D_n^w)}$
- Drawback–Increases Client storage

Verifiablity on Static SSE Schemes

Traditional Verifiable Schemes with client storage

- Target– Given w, verify $R_w = \{D_1^w, D_2^w, \dots, D_n^w\}$
- For each w, stores $\boxed{digest(w) \leftarrow H(D_1^w || D_2^w || \dots || D_n^w)}$
- Drawback–Increases Client storage

Problems of outsourcing

- Return $(digest(w_1), R_{w_1})$ for any query & gets verified correctly

Verifiablity on Static SSE Schemes

Traditional Verifiable Schemes with client storage

- Target– Given w, verify $R_w = \{D_1^w, D_2^w, \ldots, D_n^w\}$
- For each w, stores $\boxed{digest(w) \leftarrow H(D_1^w || D_2^w || \ldots || D_n^w)}$
- Drawback–Increases Client storage

Problems of outsourcing

- Return $(digest(w_1), R_{w_1})$ for any query & gets verified correctly

Our Solution for Static Verifiable SSE without extra client storage

- Bind w with the digest as $\boxed{digest(w) \leftarrow H(w || D_1^w || D_2^w || \ldots || D_n^w)}$
- Upload $\boxed{\{digest(w), D_1^w, D_2^w, \ldots, D_n^w\}}$ for all w
- Benefits: 1. Verification very efficient 2. only a hash computation
- Public verifiability not needed

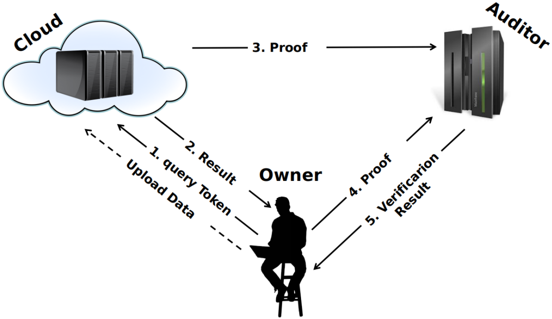
System Model

1. Init Phase

2. Search Phase

3.Update Phase

System Model

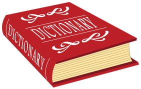


Black-box forward secure DSSE scheme Σ_f

- We take a forward secure DSSE scheme Σ_f as blackbox
- Σ_f has three phases
- We don't change anything in Σ_f

Black-box forward secure DSSE scheme Σ_f

- We take a forward secure DSSE scheme Σ_f as blackbox
- Σ_f has three phases
- We don't change anything in Σ_f
- We add an independent extra (Table) data structure T for verification

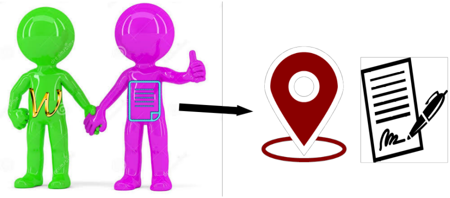


Data Structure Type

With ability to search efficiently

Our Generic Scheme

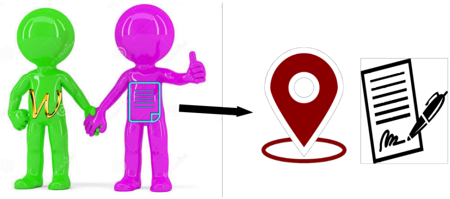
Key-value pair generation



for each Keyword-document pair a position-signature pair is generated

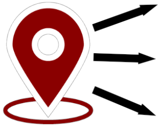
Our Generic Scheme

Key-value pair generation



for each Keyword-document pair a position-signature pair is generated

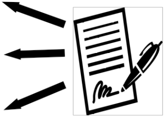
Bindings



keyword w

document Identifier id_i^w

position i



denoted as σ_i^w denoted as pos_i^w

Binding in Signature and positions

Signature Generation

signature

$\sigma_i^w = \mathcal{S}.\texttt{Sign}(sk, m_i^w)$

$m_i^w \leftarrow r_i^w.id_i^w \mod q$

binding with id_i^w

$r_i^w \leftarrow R(s_w || i)$

binding with i

$s_w \leftarrow F(K_s, w)$

binding with w

Binding in Signature and positions

Signature Generation

signature

$\sigma_i^w = \mathcal{S}.\texttt{Sign}(sk, m_i^w)$

$m_i^w \leftarrow r_i^w.id_i^w \mod q$
binding with id_i^w

$r_i^w \leftarrow R(s_w||i)$
binding with i

$s_w \leftarrow F(K_s, w)$
binding with w

Position Generation

position

$pos_i^w \leftarrow F(tag_w, id_i^w||i)$

$tag_w \leftarrow F(K_t, w)$
binding with w

id_i^w
binding with id_i^w

i
binding with i

Search

During search

1. Cloud first gets $\boxed{R_w = \{id_1^w, id_2^w, \ldots, id_n^w\}}$ using Σ_f
2. Then calculates positions $\boxed{\{pos_1^w, pos_2^w, \ldots, pos_n^w\}}$ where $pos_i^w \leftarrow F(tag_w, id_i^w || i)$
3. tag_w is provided to cloud with search token
4. Retrieve signatures as $\boxed{\{\sigma_1^w = T[pos_1^w], \sigma_2^w = T[pos_2^w], \ldots, \sigma_n^w = T[pos_n^w]\}}$

Observation

- $\prod_1^n \sigma_i^w = \prod_1^n \mathcal{S}.\mathtt{Sign}(sk, m_i^w) = \prod_1^n (g^{sk})^{m_i^w} = (g^{sk})^{\sum_1^n m_i^w}$

Verification

- We observe $\prod_{1}^{n} \sigma_i^w = (g^{sk})^{\sum_1^n m_i^w}$
- Cloud computes $\sigma_w = \prod_{1}^{n} \sigma_i^w$, Aggregate Signature
- Client computes $m = \sum_{1}^{n} m_i^w = \sum_{1}^{n} r_i^w . id_i^w$ Aggregate of IDs
- Possible as client can regenerate r_i^w and gets id_i^ws from cloud

Verification

- $\mathcal{S}.\texttt{Verify}(pk, m, \sigma_w)$

Correctness

- $\hat{e}(\mathcal{H}(m), pk) = \hat{e}(g^m, g^{sk}) = \hat{e}(g^{sk \sum m_i}, g) = \hat{e}(\prod g^{sk.m_i}, g) = \hat{e}(\prod \sigma_i, g) = \hat{e}(\sigma, g)$

Update and Forward Privacy

Assumption

- Any Σ_f stores keyword frequency list C

Update

- Owner can compute position-signature pairs for each word-doc pair
- Possible because C gives frequency
- C is also gets updated

Forward Privacy

- Cloud don't know which *id* can be added next
- $\implies$ Cant calculate positions
- $\implies$ from position, cant link with keyword

Security

Simulation

- Takes the simulator of the black-box scheme
- Additionally Simulates T_{sig}
- Simulates Query tokens.
- Simulates Update tokens.

Security

Simulation

- Takes the simulator of the black-box scheme
- Additionally Simulates T_{sig}
- Simulates Query tokens.
- Simulates Update tokens.

Indinguishibility for T_{sig}

- Indinguishibility between $g^{\alpha mr}$ and $g^{\alpha r'}$
- $r \leftarrow$ cryptographically secure pseudo-random
- $r' \leftarrow$ randomly taken

Security

Simulation

- Takes the simulator of the black-box scheme
- Additionally Simulates T_{sig}
- Simulates Query tokens.
- Simulates Update tokens.

Indinguishibility for T_{sig}

- Indinguishibility between $g^{\alpha m r}$ and $g^{\alpha r'}$
- $r \leftarrow$ cryptographically secure pseudo-random
- $r' \leftarrow$ randomly taken

Indinguishibility for Query and Update token

- Relies on Random oracle model
- The MAC function F is generated simulated with a random oracle.

Our Stands in the Literature

Table: 2: Comparison of verifiable dynamic SSE schemes

Scheme Name	Forward privacy	Publicly verifiable?	Extra Storage		Extra Computation		Extra Communication
			owner	cloud	owner	cloud	owner
Yoneyama and Kimura [YK17]	✓	×	$O(\|\mathcal{W}\|)$	$O(\|\mathcal{W}\| log \|\mathcal{DB}\|)$	$O(\|R_w\|)$	$O(\|R_w\|)$	$O(1)$
Bost and Fouque [BFP16]	×	×	$O(\|\mathcal{W}\|)$	$O(\|\mathcal{W}\|)$	$O(\|R_w\|)$	$O(1)$	$O(1)$
Miao et al. [MWWM19]	×	✓	$O(\|\mathcal{W}\|)$	$O(N + \|\mathcal{W}\|)$	$O(\|R_w\|)$	$O(\|R_w\|)$	$O(1)$
Zhu et al. [ZLW16]	×	×	$O(1)$	$O(1)$	$O(\|R_w\|)$	$O(\|R_w\| + N)$	$O(\|R_w\|)$
Jiang et al. [JZGL15]	×	✓	$O(1)$	$O(\|\mathcal{W}\|)$	$O(\log \|\mathcal{W}\|)$	$O(\|R_w\| + N)$	$O(1)$
Ψ_f	✓	✓	$O(1)$	$O(N)$	$O(\|R_w\|)$	$O(\|R_w\|)$	$O(1)$

Where N is the #keyword-doc pairs.

Results

- Ψ_f is very efficient with respect to low resource owner.
- To verify the search, owner needs only $|R_w|$ multiplication which very less from the others.
- The owner also does not require any extra storage

Possible future works

Increasing Efficiency

- Storage reduction
- Communication cost reduction
- Computation cost reduction

Verifiability on Complex Queries

- Conjunctive or Boolean Queries
- Ranked search
- Range searched

Verifiablity with more secure schemes

- Backward secure schemes
- Type-I, Type-II and Type-III backward Secrecy

Raphael Bost, Pierre-Alain Fouque, and David Pointcheval.
Verifiable dynamic symmetric searchable encryption: Optimality and forward security.
IACR Cryptology ePrint Archive, 2016:62, 2016.

Qi Chai and Guang Gong.
Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers.
In *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*, pages 917–922, 2012.

Rong Cheng, Jingbo Yan, Chaowen Guan, Fangguo Zhang, and Kui Ren.
Verifiable searchable symmetric encryption from indistinguishability obfuscation.
In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015*, pages 621–626, 2015.

Shunrong Jiang, Xiaoyan Zhu, Linke Guo, and Jianqing Liu.
Publicly verifiable boolean query over outsourced encrypted data.
In *2015 IEEE Global Communications Conference, GLOBECOM 2015, San Diego, CA, USA, December 6-10, 2015*, pages 1–6, 2015.

Zheli Liu, Tong Li, Ping Li, Chunfu Jia, and Jin Li.
Verifiable searchable encryption with aggregate keys for data sharing system.
Future Generation Comp. Syst., 78:778–788, 2018.

Yuxi Li, Fucai Zhou, Yuhai Qin, Muqing Lin, and Zifeng Xu.
Integrity-verifiable conjunctive keyword searchable encryption in cloud storage.
Int. J. Inf. Sec., 17(5):549–568, 2018.

Meixia Miao, Jianfeng Wang, Sheng Wen, and Jianfeng Ma.
Publicly verifiable database scheme with efficient keyword search.
Inf. Sci., 475:18–28, 2019.

Wakaha Ogata and Kaoru Kurosawa.
Efficient no-dictionary verifiable searchable symmetric encryption.
In *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, pages 498–516, 2017.

Azam Soleimanian and Shahram Khazaei.
Publicly verifiable searchable symmetric encryption based on efficient cryptographic components.
Des. Codes Cryptography, 87(1):123–147, 2019.

Jianfeng Wang, Xiaofeng Chen, Shifeng Sun, Joseph K. Liu, Man Ho Au, and Zhi-Hui Zhan.
Towards efficient verifiable conjunctive keyword search for large encrypted database.
In *Computer Security - 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part II*, pages 83–100, 2018.

Cheng Xu, Ce Zhang, and Jianliang Xu.
vchain: Enabling verifiable boolean range queries over blockchain databases.
CoRR, abs/1812.02386, 2018.

Kazuki Yoneyama and Shogo Kimura.
Verifiable and forward secure dynamic searchable symmetric encryption with storage efficiency.
In *Information and Communications Security - 19th International Conference, ICICS 2017, Beijing, China, December 6-8, 2017, Proceedings*, pages 489–501, 2017.

Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou.
All your queries are belong to us: The power of file-injection attacks on searchable encryption.
In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 707–720, 2016.

Xiaoyu Zhu, Qin Liu, and Guojun Wang.
A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing.
In *2016 IEEE Trustcom/BigDataSE/ISPA, Tianjin, China, August 23-26, 2016*, pages 845–851, 2016.

Questions?

Thank You!

Leakage Function

- $\mathcal{L}^{\Psi_f}_{bld}(\mathcal{DB}) = \{\mathcal{L}^{\Sigma_f}_{bld}(\mathcal{DB}), |T_{sig}|\}$
- $\mathcal{L}^{\Psi_f}_{srch}(w) = \{\mathcal{L}^{\Sigma_f}_{srch}(w), \{(id^w_i, pos^w_i, \sigma^w_i) : i = 1, 2, \ldots, c_w\}\}$
- $\mathcal{L}^{\Psi_f}_{updt}(f) = \{id, \{(\mathcal{L}^{\Sigma_f}_{updt}(w_i, id), pos^{w_i}, \sigma^{w_i}) : i = 1, 2, \ldots, n_{id}\}\}$

Related Works

Type 1 Works

- Should be added

Type 2 Works

- Should be added

Type 3 Works

- Should be added