Understanding and Mitigating Leakage-Abuse Attacks against Searchable Encryption

Raphael Bost[1], Pierre-Alain Fouque[2], Brice Minaud[3]

[1]Direction Générale de l'Armement - Maîtrise de l'Information

[2]Université de Rennes 1

[3]INRIA & Ecole Normale Supèrieure

ICERM's Encrypted Search Workshop
06/10/2019
Providence, RI

Disclaimers

- These slides have been made very recently (like in finished last night).
- Jetlag

Disclaimers

- These slides have been made very recently (like in finished last night).
- Jetlag
- Support for a discussion: please ask questions.
  If you see something, say something.

Disclaimers

- These slides have been made very recently (like in finished last night).
- Jetlag
- Support for a discussion: please ask questions.
  If you see something, say something.

Claim

These are the (maybe) controversial points.

Security Definition

Indistinguishability-based security definition [CGKO06] (in a general form).

```
Init($\mathsf{DB}^0, \mathsf{DB}^1$)
    if $\mathcal{L}^{\mathsf{Stp}}(\mathsf{DB}^0) \neq \mathcal{L}^{\mathsf{Stp}}(\mathsf{DB}^1)$
        Abort game
    $b \xleftarrow{\$} \{0,1\}$
    $(\mathsf{EDB}, K_\Sigma, \sigma) \xleftarrow{\$} \mathsf{Setup}(\mathsf{DB}^b)$
    return EDB
Final($b'$)
    return $b = b'$
```

```
Query($q_i^0, q_i^1$)
    if $\mathcal{L}^{\mathsf{Query}}(q_i^0) \neq \mathcal{L}^{\mathsf{Query}}(q_i^1)$
        Abort game
    $(R, \sigma, \tau; \mathsf{EDB}) \xleftarrow{\$} \mathsf{Query}(K_\Sigma, \sigma, q_i^b; \mathsf{EDB})$
    return $\tau$
```

The sequence $(\mathsf{DB}, q_1, \ldots, q_n)$ is called an *history*.

Leakage-Abuse Attacks

- Introduced as *inference attack* in [IKK12]: use co-occurrence information against an encrypted DB.
- Improved in [CGPR15] : combine co-occurrence with the volume leakage.

Leakage-Abuse Attacks

- Introduced as *inference attack* in [IKK12]: use co-occurrence information against an encrypted DB.
- Improved in [CGPR15] : combine co-occurrence with the volume leakage.
- Exploit the scheme's leakage to attack the DB or the queries.

Leakage-Abuse Attacks

These attacks have many variants:

- Against DB supporting range queries [KKNO16, GLMP19]
- Against DB supporting k-nearest-neighbor [KPT19]
- Against dynamic DB: file injection attacks [ZKP16]

Leakage-Abuse Attacks

These attacks have assume the adversary has some auxiliary information:

- [IKK12]: distribution of the co-occurrence database
- [CGPR15]: co-occurrence + keyword distribution
- [KKNO16]: queries are uniformly distributed
- [ZKP16]: knowledge of the adversarially inserted documents

Also, you almost always achieve 100% reconstruction of the database/queries.

Leakage-Abuse Attacks

Why do they work ?

The security definition should cover these attacks. . .

The model guarantees that two executions of a SE scheme cannot be distinguished; LAAs retrieve the database or the queries.

Leakage-Abuse Attacks

Why do they work ?

The security definition should cover these attacks. . .

The model guarantees that two executions of a SE scheme cannot be distinguished; LAAs retrieve the database or the queries.

Claim

In these attacks, the observed leakage is conditioned to some additional knowledge by the adversary. The combination of both can uniquely identify a history.

Singular histories

An history H such that there is no other history $H' \neq H$ with $\mathcal{L}(H) = \mathcal{L}(H')$ is call *singular* [CGKO06]. For singular histories, the ind-based security definition becomes void.

> *Note that the existence of a second history with the same trace is a necessary assumption, otherwise the trace would immediately leak all information about the history.*

Singular histories: examples

- In [IKK12, CGPR15], the adversary 'chooses' the database. It is impossible to find two lists of queries with the same leakage with this database.
- In [KKNO16], the adversary knows that the queries are uniformly distributed. It is impossible to find two databases with the same volume leakage.

Singular histories: examples

- In [IKK12, CGPR15], the adversary 'chooses' the database. It is impossible to find two lists of queries with the same leakage with this database.
- In [KKNO16], the adversary knows that the queries are uniformly distributed. It is impossible to find two databases with the same volume leakage.

Claim

The security definition protect that database and *all* the queries *as a whole*, not in isolation.

LAAs against other security definitions

LAAs are not restricted to SE: leakage applies to other types of encryption:

- CPA/CCA encryption 'leaks' the size of the message. The length of messages is a very useful information when attacking encrypted traffic [SSV12] => TFC.
- Functional encryption 'leaks' the result of the function evaluation. (Non-adaptive) SE security can be seen as a restriction of (non-adaptive) functional encryption security.

LAAs against other security definitions

Consider the following example: define an encryption scheme on a message space $\mathcal{M}$ such that $\forall m \neq m' \in \mathcal{M}, |m| \neq |m'|$. The encryption/decryption algorithm is the identity function: $\mathrm{Enc}(m) = m$.

Strictly speaking, this scheme is CPA secure: $\forall m, m' \in \mathcal{M}$ s.t. $|m| = |m'|, \mathrm{Enc}(m) = \mathrm{Enc}(m')$.

LAAs against other security definitions

Consider the following example: define an encryption scheme on a message space $\mathcal{M}$ such that $\forall m \neq m' \in \mathcal{M}, |m| \neq |m'|$. The encryption/decryption algorithm is the identity function: $\mathrm{Enc}(m) = m$.

Strictly speaking, this scheme is CPA secure: $\forall m, m' \in \mathcal{M}$ s.t. $|m| = |m'|, \mathrm{Enc}(m) = \mathrm{Enc}(m')$.

Claim

In other security definitions, there are constrains that prevent the definition to turn out void.

Constraints

We need a formalization of auxiliary information available to the adversary: an history *conforms* to some constraints (*i.e.* is compatible with prior adversarial knowledge).

Constraints

We need a formalization of auxiliary information available to the adversary: an history *conforms* to some constraints (*i.e.* is compatible with prior adversarial knowledge).

Definition (Constraint)

A *constraint* C is a predicate over the set of all possible histories. A history H is said to *satisfy* the constraint C if and only if $C(H) = \text{true}$. It is valid if $\exists H \neq H', C(H) = C(H') = \text{true}$.

Resilience

For a given constraint (representing adversarial knowledge), the leakage of a scheme should not uniquely identify the history.

Definition (Resilience)

A leakage function $\mathcal{L}$ is *resilient* to the constraint C iff for every history H satisfying C, there exists a distinct history $H' \neq H$ satisfying C such that $\mathcal{L}(H') = \mathcal{L}(H)$.
If $\mathfrak{C}$ is a set of constraints, $\mathcal{L}$ is said to be *resilient* to $\mathfrak{C}$ iff it is resilient to all $C \in \mathfrak{C}$.

This already precludes most of the leakage-abuse attacks discussed previously.

Examples of Constraints: knowledge of the DB

How to capture the prior knowledge of the database?

$$C^{\widetilde{\mathrm{DB}}}(H) = C^{\widetilde{\mathrm{DB}}}(\mathrm{DB}, q_1, \dots) = \mathrm{true} \Leftrightarrow \mathrm{DB} = \widetilde{\mathrm{DB}}$$

$$\mathfrak{C}^{\mathcal{DB}} = \{C^{\mathrm{DB}}, \mathrm{DB} \in \mathcal{DB}\}$$

From [CGPR15], $L1$ is not resilient to $C^{\widetilde{\mathrm{DB}}}$ for any $\widetilde{\mathrm{DB}}$.

Examples of Constraints: known document subset

$$C^{D_1,\ldots,D_\ell}(H) = \text{true} \Leftrightarrow D_1, \ldots, D_\ell \in \text{DB}$$

[CGPR15]: $L3$ (keyword occurrences) is not resilient to $C^{D_1,\ldots,D_\ell}$.

Examples of Constraints: file injections

The constraint C associated to an adversary who injects the documents $D_1, \ldots, D_\ell$ at queries $i_1, \ldots, i_\ell$ is true iff $\forall 1 \leq j \leq \ell, q_{i_j}$ is an update query inserting D_j.

[ZKP16]: the search pattern leakage is not resilient to leakage injection constraints.

Stronger forms of resilience

The resilience definition gives us a very weak form of security: the choice between two histories.

Definition (α-resilience)

A leakage function $\mathcal{L}$ is *α-resilient* to the constraint C iff for every history H satisfying C, there exist α pairwise distinct histories $(H_i)_{i \leq \alpha}$ satisfying C such that $\forall i, \mathcal{L}(H_i) = \mathcal{L}(H)$.
If $\mathfrak{C}$ is a set of constraints, $\mathcal{L}$ is said to be *α-resilient* to $\mathfrak{C}$ iff it is α-resilient to all $C \in \mathfrak{C}$.

Stronger forms of resilience

α-resilience is still not enough: all the α histories can be identical on most of the queries – the notion does not cover partial reconstruction.

Definition (α-resilience per query)

A leakage function $\mathcal{L}$ is *α-resilient per query* to the constraint C iff for every history $H = (\mathrm{DB}, q_1, \ldots, q_n)$ satisfying C, and every $i \in [1, n]$, there exist α pairwise distinct histories $(H_j)_{j \leq \alpha}$ differing from H only at the i-th query, satisfying C, and such that $\forall j, \mathcal{L}(H_j) = \mathcal{L}(H)$.

Achieving resilience

We need tools to show the resilience of a leakage function with respect to some constraints.
Suppose the leakage $\mathcal{L}$ is s.t. $\mathcal{L}(q) = f(\mathsf{DB}, q)$ (*e.g.* volume leakage). Then, if H, $H||q$ and $H||q'$ satisfy C, and $f(\mathsf{DB}, q) = f(\mathsf{DB}, q')$, then, $H||q$ and $H||q'$ are two histories with the same leakage satifying C.

We can constructively and iteratively construct many histories satisfying the constraint, with the same leakage, and thus prove resilience.

Achieving resilience

We can regroup keywords according to the value of $f(\mathsf{DB}, \cdot)$

$$\begin{aligned}\Gamma_{\mathcal{L}}(H) &= \{\{q \in \mathcal{Q} : f(\mathsf{DB}, q) = \ell\} : \ell \in \mathrm{Im}(f)\} \\ &= \{G_1, \ldots, G_m\}\end{aligned}$$

Achieving resilience

We can regroup keywords according to the value of $f(\mathrm{DB}, \cdot)$

$$\begin{aligned}\Gamma_{\mathcal{L}}(H) &= \{\{q \in \mathcal{Q} : f(\mathrm{DB}, q) = \ell\} : \ell \in \mathrm{Im}(f)\} \\ &= \{G_1, \ldots, G_m\}\end{aligned}$$

Claim

$\mathcal{L}$ is α-query-resilient with $\alpha = \min |G_i|$

Achieving resilience for length leakage

- $f(\mathrm{DB}, w) = |\mathrm{DB}(w)|$
- With padding, $f(\mathrm{DB}, w) = |\mathrm{DB}(w)| + p(w)$
- Construct p such that it forms large clusters:

$$\forall w, \left|\{w' \text{ s.t. } |\mathrm{DB}(w)| + p(w) = |\mathrm{DB}(w')| + p(w')\}\right| \geq \alpha$$

- We also want to minimize the cost $\sum_{w} p(w)$

Achieving resilience for length leakage

- This is an optimization problem, that can be solved in $\mathcal{O}(\alpha K)$ time and $\mathcal{O}(K)$ memory.
- This approach can be applied to hide the communication volume on a secure channel at an optimal cost.
- It can be adapted to dynamic databases, with distributional knowledge from the adversary.

Achieving resilience for length leakage – variant

What happens when the query distribution is not uniform? Then, α-resilience as defined previously is not sufficient: for a given leakage, one query might be much more likely than the $\alpha - 1$ others. The min-entropy of the query distribution must be lower bounded by $\log_2 \alpha$.

Achieving resilience for length leakage – variant

What happens when the query distribution is not uniform? Then, α-resilience as defined previously is not sufficient: for a given leakage, one query might be much more likely than the $\alpha - 1$ others. The min-entropy of the query distribution must be lower bounded by $\log_2 \alpha$.

Claim

The resilience notion can be transformed to support distributional knowledge (*i.e.* distributional constrains).

Achieving resilience for length leakage – variant

In the case of length leakage, is it possible to find an optimal padding according to a query distribution? Is it possible to use different cost functions (others than the total storage cost) and find an optimal padding according to this cost function?

Achieving resilience for length leakage – variant

In the case of length leakage, is it possible to find an optimal padding according to a query distribution? Is it possible to use different cost functions (others than the total storage cost) and find an optimal padding according to this cost function?

Claim

Trying to find optimum padding in the general case is NP-complete. If $\mathrm{P} \neq \mathrm{NP}$, it is not in APX.

Conclusion

- LAAs are super important for the field when assessing the actual security of schemes.

Conclusion

- LAAs are super important for the field when assessing the actual security of schemes.
- For a given leakage the actual security depends a lot on the adversary's prior knowledge.

Conclusion

- LAAs are super important for the field when assessing the actual security of schemes.
- For a given leakage the actual security depends a lot on the adversary's prior knowledge.
- We can construction definitions that take this fact into account.

Conclusion

- LAAs are super important for the field when assessing the actual security of schemes.
- For a given leakage the actual security depends a lot on the adversary's prior knowledge.
- We can construction definitions that take this fact into account.
- For some cases, we can improve the practical security of schemes at a reduced cost.

Conclusion

- LAAs are super important for the field when assessing the actual security of schemes.
- For a given leakage the actual security depends a lot on the adversary's prior knowledge.
- We can construction definitions that take this fact into account.
- For some cases, we can improve the practical security of schemes at a reduced cost.
- In general the security guarantees are weak or hard to achieve.

Questions?

References I

- Reza Curtmola, Juan A. Garay, Seny Kamara, and Rafail Ostrovsky, *Searchable symmetric encryption: improved definitions and efficient constructions*, ACM CCS 2006 (Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, eds.), ACM Press, October / November 2006, pp. 79–88.
- David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart, *Leakage-abuse attacks against searchable encryption*, ACM CCS 2015 (Indrajit Ray, Ninghui Li, and Christopher Kruegel, eds.), ACM Press, October 2015, pp. 668–679.

References II

- Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth G. Paterson, *Learning to reconstruct: Statistical learning theory and encrypted database attacks*, IEEE Symposium on Security and Privacy (S&P) 2019, 2019.
- Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu, *Access pattern disclosure on searchable encryption: Ramification, attack and mitigation*, NDSS 2012, The Internet Society, February 2012.

References III

- Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'Neill, *Generic attacks on secure outsourced databases*, ACM CCS 2016 (Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, eds.), ACM Press, October 2016, pp. 1329–1340.

- Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia, *Data recovery on encrypted databases with k-nearest neighbor query leakage*, IEEE Symposium on Security and Privacy (S&P) 2019, 2019.

References IV

- Ahmad-Reza Sadeghi, Steffen Schulz, and Vijay Varadharajan, *The silence of the LANs: Efficient leakage resilience for IPsec VPNs*, ESORICS 2012 (Sara Foresti, Moti Yung, and Fabio Martinelli, eds.), LNCS, vol. 7459, Springer, Heidelberg, September 2012, pp. 253–270.
- Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou, *All your queries are belong to us: The power of file-injection attacks on searchable encryption*, USENIX Security 2016 (Thorsten Holz and

References V

Stefan Savage, eds.), USENIX Association, August 2016, pp. 707–720.