

Forward & Backward Private Searchable Encryption from Constrained Cryptographic Primitives

Raphael Bost, Brice Minaud, Olga Ohrimenko

Great Co-Authors



*Brice Minaud
RHUL*



*Olga Ohrimenko
MSR Cambridge*

Searchable Encryption

Outsource data

- Securely
- Keep search functionalities
- Aimed at efficiency
- ... we have to leak some information

Forward Privacy

- Forward-private: an update does not leak any information on the updated keywords (often, no information at all)
- Thwart adaptive file injection attacks [ZKP16]
- Few existing constructions
 - [SPS14]: ORAM-based, expensive updates
 - [B16]: Asymptotically optimal, (very) low update throughput in practice

A Simple Dynamic Scheme

- In regular index-based schemes: suppose w matches $DB(w) = (ind_1, \dots, ind_n)$.

$$K_w || K'_w \leftarrow H(K, w)$$

$$\forall 1 \leq i \leq n_w, t_i \leftarrow F(K_w, i), EDB[t_i] \leftarrow F(K'_w, i) \oplus ind_i$$

Search(w): the client sends $F(K, w)$ to the server

- *Update(add, w, ind)*: Client computes $t_{n+1} \leftarrow F(K_w, n_w + 1)$, $c \leftarrow F(K'_w, n_w + 1) \oplus ind_i$, sends (t_{n+1}, c)
- Not forward-private: the server can compute t_{n+1} from K_w

Constrained PRF

- Can we restrict the evaluation of $F(K_w, \cdot)$ on $[1, n]$?
- $\text{PRF}: \text{Setup} \rightarrow K$ $\text{Eval}(K, x) \rightarrow F(K, x)$
- $\text{CPRF}: \text{Setup} \rightarrow K$ $\text{Constrain}(K, C) \rightarrow K_C$
 $\text{Eval}(K_C, x) \rightarrow F(K, x) \text{ if } C(x) = 1, \perp \text{ otherwise}$
- $F(K, x)$ is indistinguishable from random as long as no K_C with $C(x)=1$ has been released
- Introduced in [BW13], [KPTZ13], and [BGI14]
Many applications (e.g. broadcast encryption)

Range-Constrained PRF

- Consider the circuits $C_n(x) = 1$ if and only if $1 \leq x \leq n$ (range circuits)
- $K^n = \text{Constrain}(K, n)$ can only be used to evaluate F on $[1, n]$

Generic FP from Range-Constrained PRF (FS-RCPRF)

- $K_w || K'_w \leftarrow H(K, w)$
 $\forall 1 \leq i \leq n, t_i \leftarrow F(K_w, i), EDB[t_i] \leftarrow F(K'_w, i) \oplus ind_i$ (as before)
- *Update(add, w, ind)*: Client sends
 $(t_{n+1}, c) \leftarrow (F(K_w, n+1), F(K'_w, n) \oplus ind)$ (as before)
- *Search(w)*: the client sends *Constrain*(K_w, n) to the server. The server calls *Eval*(K^n_w, x) on $1 \leq x \leq n$
- The server cannot use K^n_w to track future updates \rightarrow Forward privacy

Diana: GGM instantiation of FS-RCPRF

- Instantiate F with the tree-based PRF construction of GGM
- Asymptotically less efficient than Σοφος
- In practice, a lot better. Always IO bounded (for both searches and updates)
- Search: $<1\mu\text{s}$ per match (on RAM)
Update: 174 000 entries per second (4300 for Σοφος)



Deletions

How to [delete](#) entries in an encrypted database?

- Existing schemes use a ‘revocation list’
- Pb: the deleted information is still [revealed](#) to the server
- Backward privacy: ‘nothing’ is leaked about the deleted documents

Backward privacy

We can define several flavors of backward privacy:

- I. Backward privacy with insertion pattern
- II. Backward privacy with update pattern
- III. Weak backward privacy

Backward privacy with insertion pattern

Leaks:

- The documents currently matching w,
- When they were inserted
- The total number of updates on w

Backward privacy with update pattern

Leaks:

- The documents currently matching w,
- When they were inserted
- When all the updates (add & del) on w happened

Weak backward privacy

Leaks:

- The documents currently matching w,
- When they were inserted
- When all the updates (add & del) on w happened
- Which deletion update canceled which insertion update

Example of the differences

Consider the sequence of updates

$(+, ind_1, \{w_1, w_2\}) ; (+, ind_2, \{w_1\}) ; (-, ind_1, \{w_1\}) ; (+, ind_3, \{w_2\})$

Search(w_1) leaks:

- I. ind_2 and that it was added at time 2.
- II. Leakage for I. + w_1 updated at times 1, 2, and 3.
- III. Leakage for II. + the entry inserted at time 1 was deleted at time 3.

A baseline construction

Baseline: the client fetches the encrypted lists of inserted and deleted documents, locally decrypts and retrieves the documents.

The encrypted lists are implemented using forward-private SSE.

- ✗ 2 interactions & $O(a_w)$ communication complexity

Moneta & Fides

- Moneta: baseline construction with ORAM-based SSE
 - Backward privacy with insertion pattern
 - Very high computational and communicational cost
- Fides: baseline construction using Diana/Σοφος
 - Backward privacy with update pattern
 - Reduced cost compared to Moneta

Backward privacy with optimal updates & communication

Could we prevent the server from decrypting some entries?

- ❖ Puncturable Encryption [GM'15]: Revocation of decryption capabilities for specific messages
- ❖ Encrypt a message with a **tag**. Revoke the ability to decrypt a set of tags: **puncture** the secret key
- ❖ Based on non-monotonic **ABE** [OSW'07]

Backward privacy from Puncturable Encryption

- Insert (w, ind) : encrypt (w, ind) with tag $t = H(w, ind)$, and add it to a (possibly forward-private) SE scheme Σ
- Delete: puncture the secret key on tag $t = H(w, ind)$
- Search w : search w in Σ and give the punctured SK to the server. Server decrypts the non-deleted results.

Backward privacy from Puncturable Encryption

Pb: the punctured SK size grows linearly (# deletions). One additional key element per deletion.

- Outsource the storage: put the SK elements in a new SSE instance on the server
- Requires an incremental PE scheme (as [GM'15])
The puncture alg. only needs a constant fraction of SK

$$\text{Puncture}(\text{SK}, t) = \text{IncPunct}(\text{sk}_0, t, d) = (\text{sk}'_0, \text{sk}_d)$$

- sk_0 is stored locally

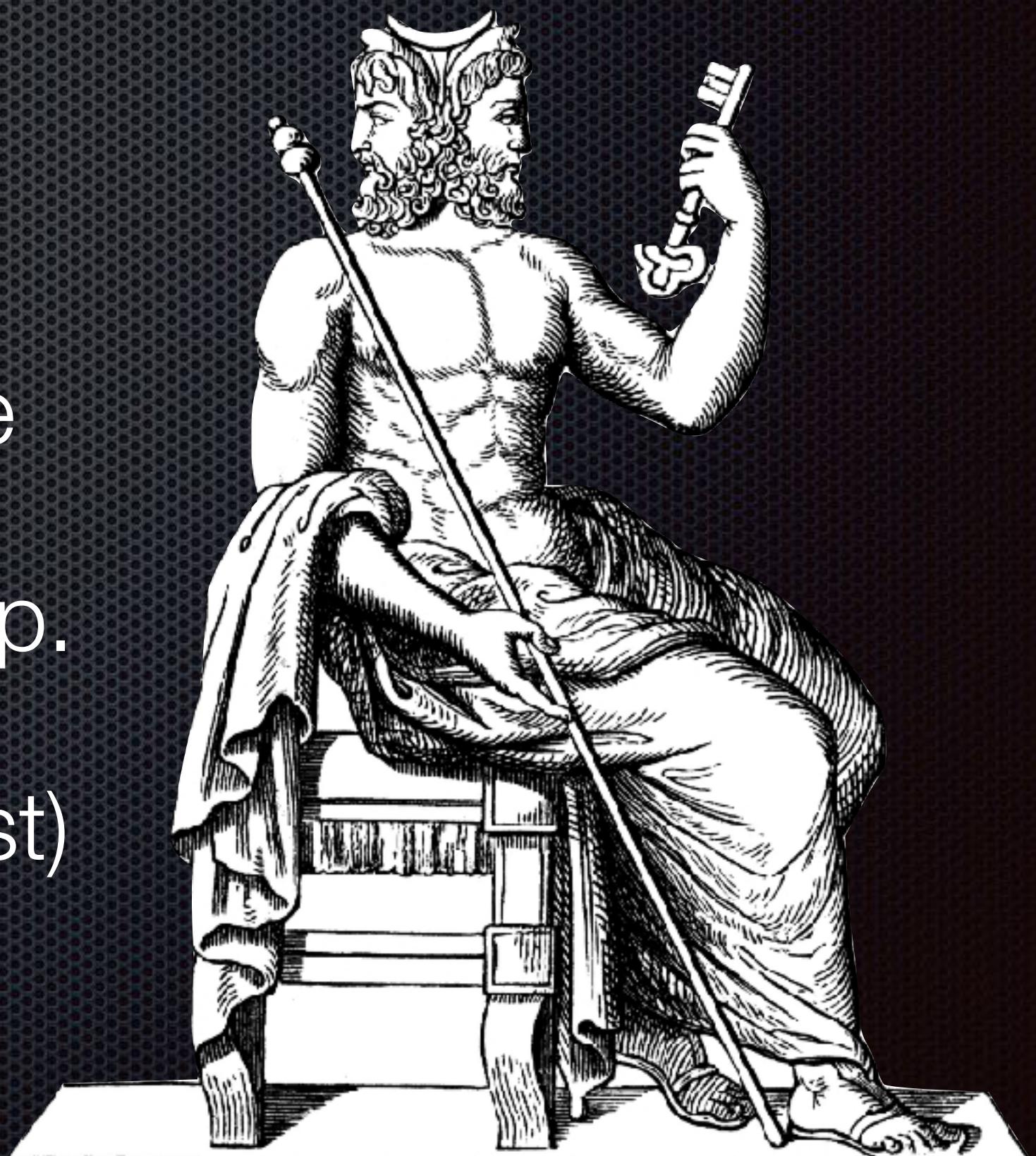
Janus

Good:

- ✓ Forward & backward-private
- ✓ Optimal update complexity
- ✓ Optimal communication

Not so good:

- ✗ $O(|W|)$ client storage
- ✗ $O(n_w \cdot d_w)$ search comp.
- ✗ Uses pairings (not fast)



Conclusion

- New way to construct forward-private schemes from constrained PRFs
- Diana: Super efficient construction made possible from CPRFs
- Definition and constructions of backward privacy
- Janus: First single roundtrip backward private construction

Questions?

ia.cr/2017/805

opensse.github.io

A circular word cloud centered on the word "OpenSeSSE". The words are arranged in concentric circles around the center. The colors of the words vary from dark blue to light blue, creating a gradient effect. Some words are partially obscured by others. The words include:

- Implementation
- Insertion
- Forward
- Security
- Cloud
- Fides
- Janus
- Moneta
- Diana
- backward
- puncturable
- leakage
- confidentiality
- constrained
- PRF
- proof
- query
- deletion
- ORAM
- privacy
- SSE
- forwards