Mylar

Building Web Applications on Top of Encrypted Data

Problem

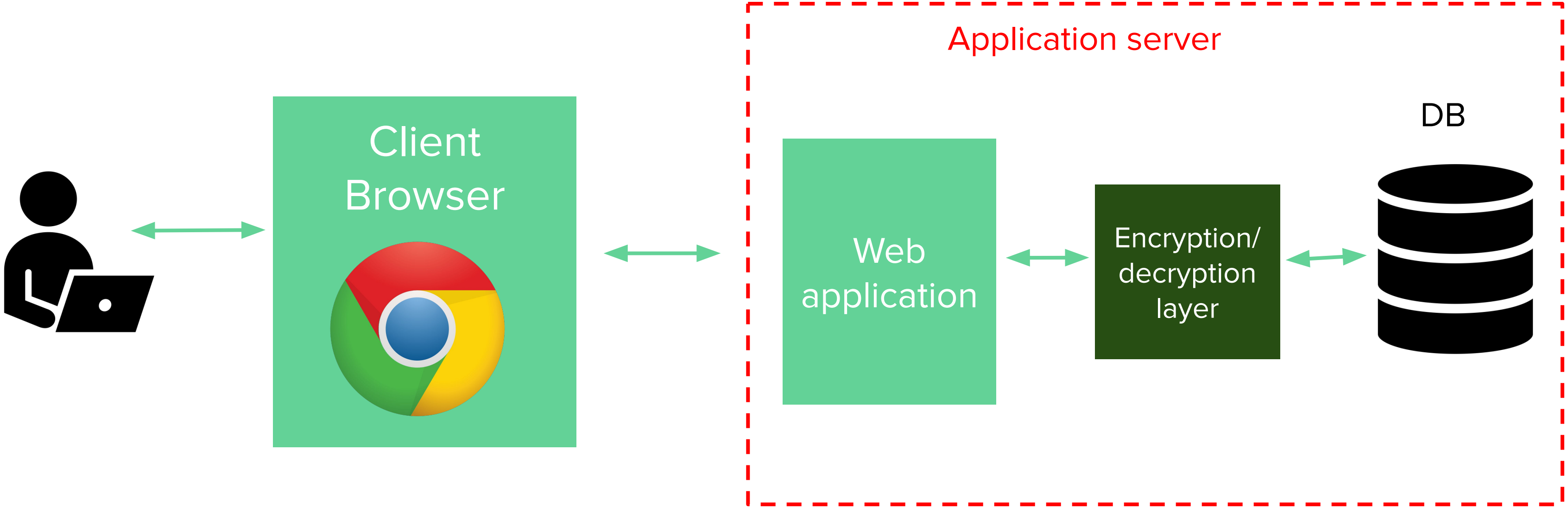
- Web applications use servers to store and process confidential information.
    - Anyone who gains access to the server can obtain all of the data stored there.

Solution

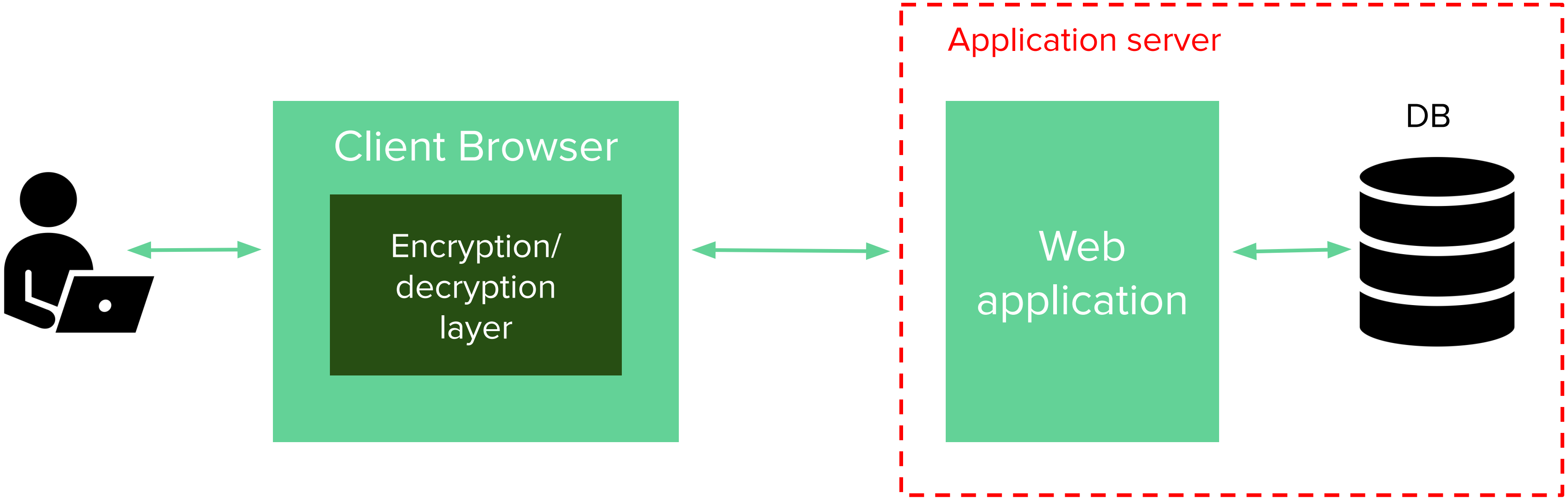
Mylar assumes malicious or compromised server operator.

1. Mylar allows users to share keys and data securely in the presence of an active adversary (man in the middle attack or a malicious administrator actively tampering with the data sent to the client)
2. Mylar allows the server to perform keyword search over encrypted documents
3. Mylar ensures that client-side application code is authentic, even if the server is malicious.

Background

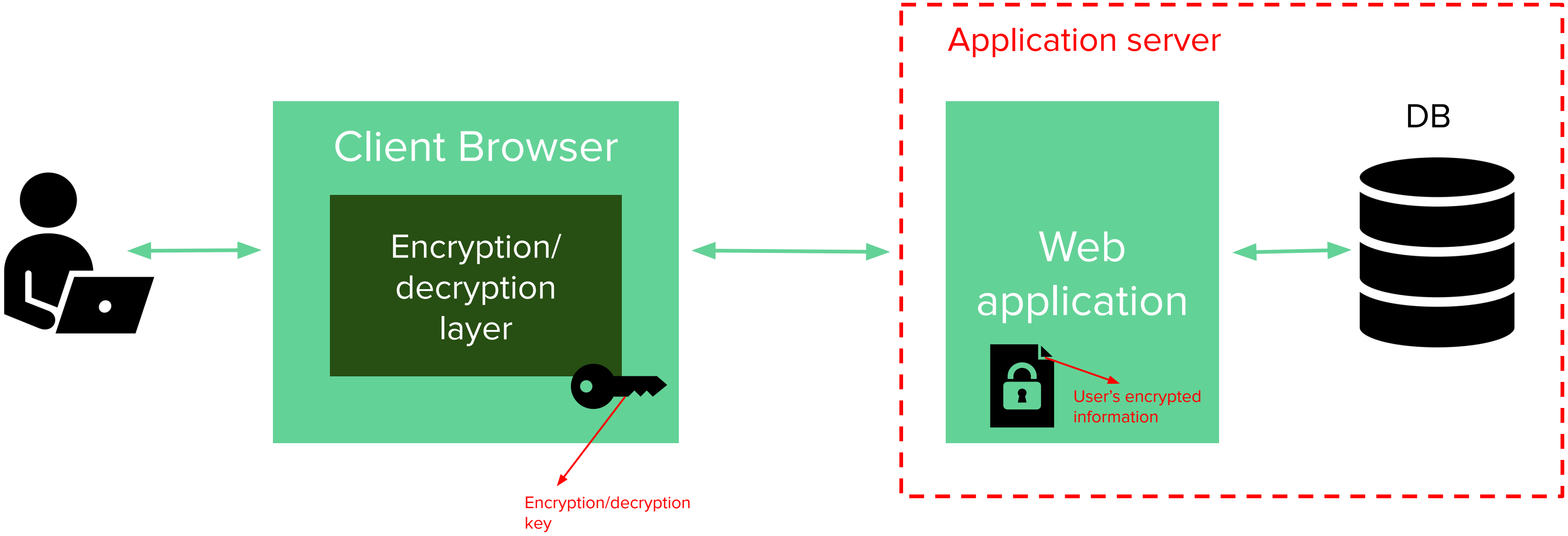

Client Browser
Application server
Web application
Encryption/ decryption layer
DB

Mylar’s model



* Assumes that site owner is non-malicious.

Mylar’s model


Client Browser
Encryption/ decryption layer
Encryption/decryption key
Application server
Web application
User’s encrypted information
DB

* Assumes that site owner is non-malicious.

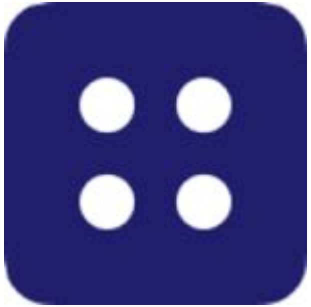
Question

- What does this design remind you of?

Question

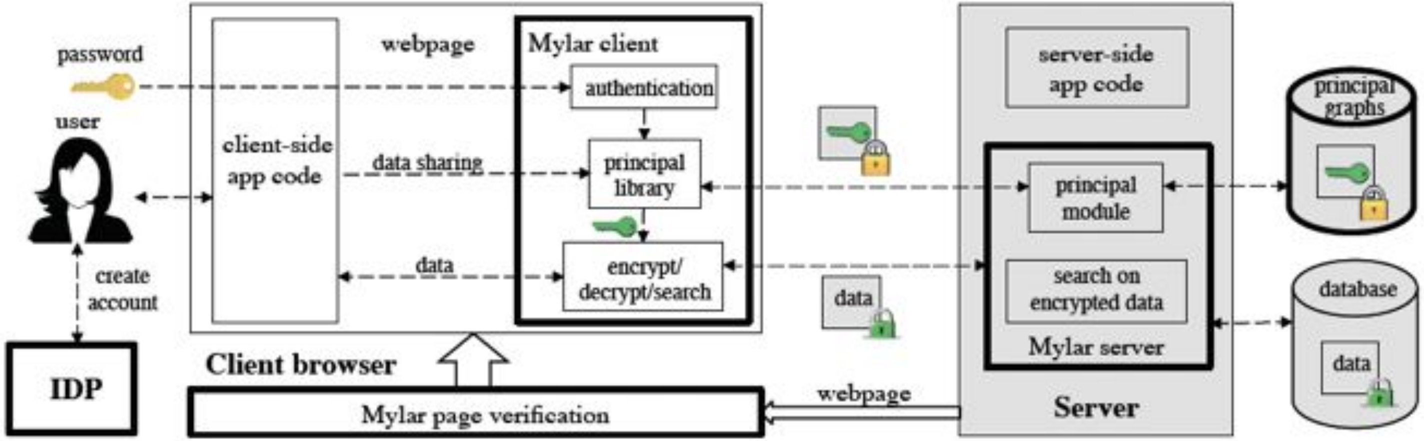
- What does this design remind you of?

Here's a hint...



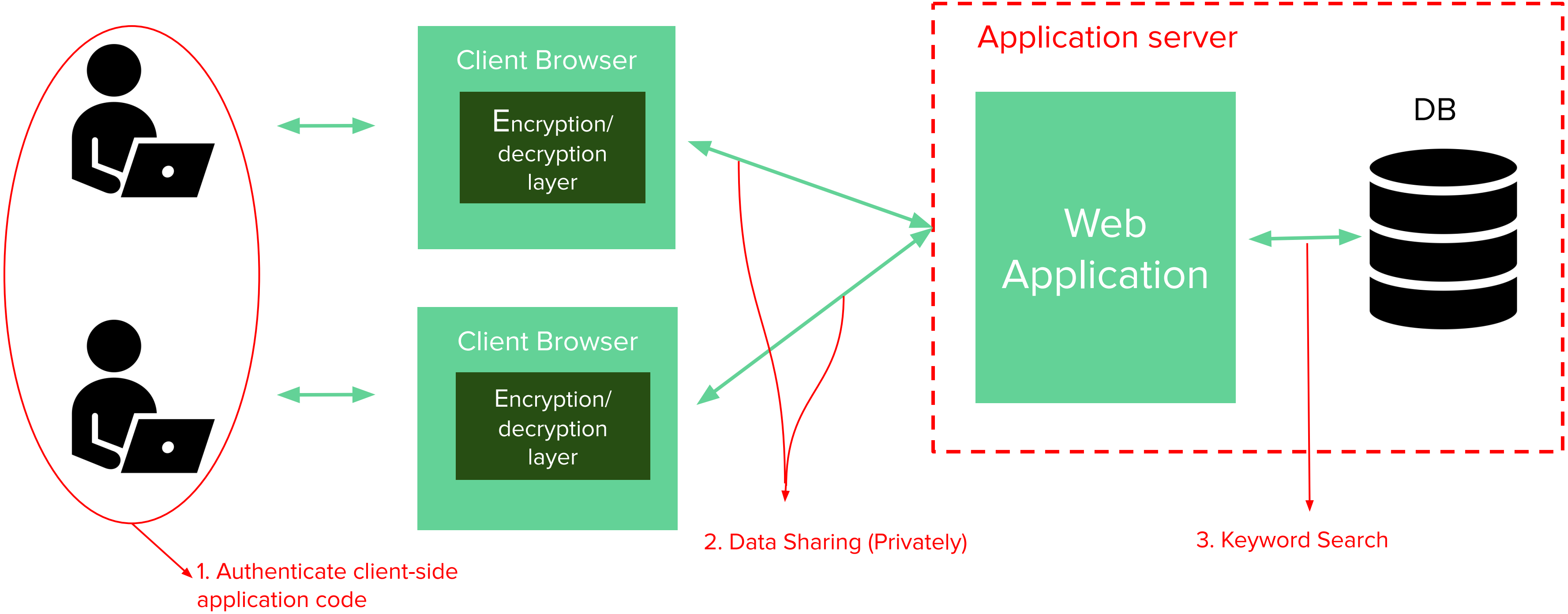
Architecture

- **Browser extension**: Verify that the code of application has not been tampered with.
- **Client-side library**: Intercepts data sent to and from the server, and encrypts or decrypts that data.
- **Server-side library**: Performs computation over encrypted data at the server.
- **IDP**: Verify that a given public key belongs to a particular username.



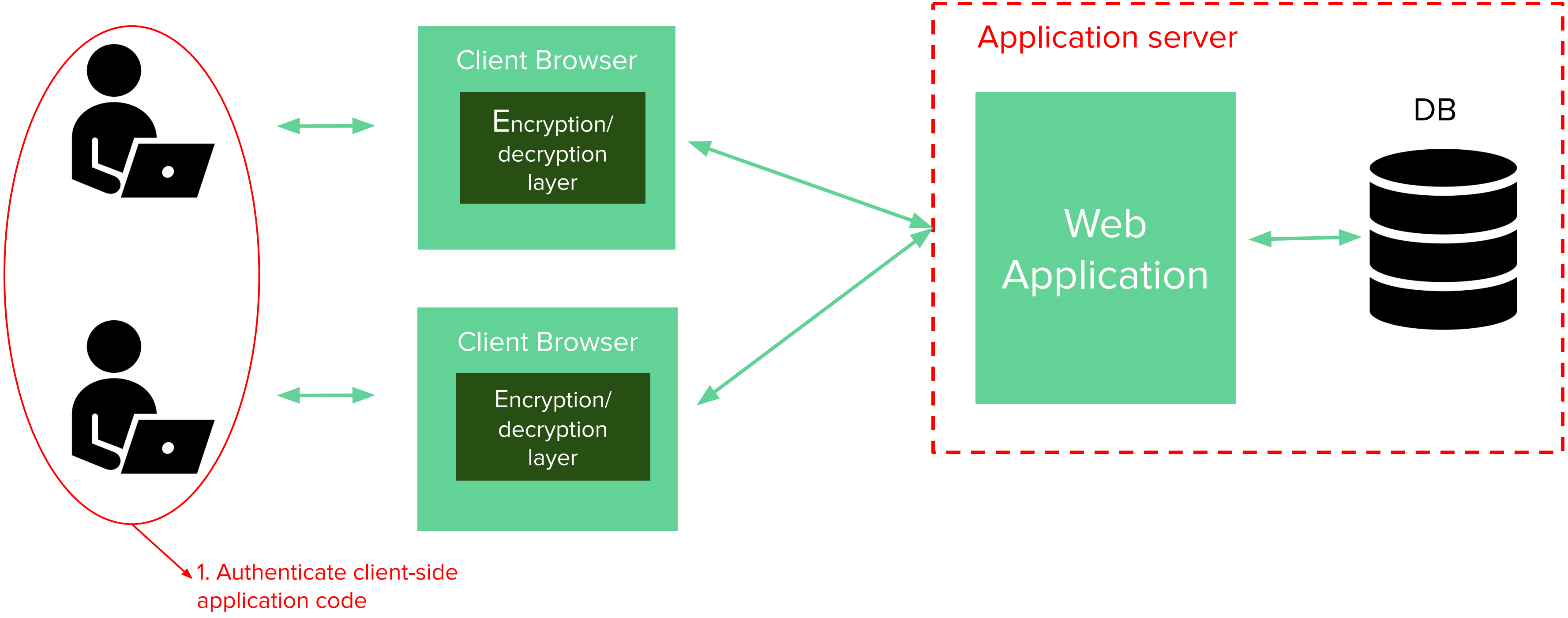
*Mylar assumes that IDP correctly verifies a users identity

Mylar’s threat model

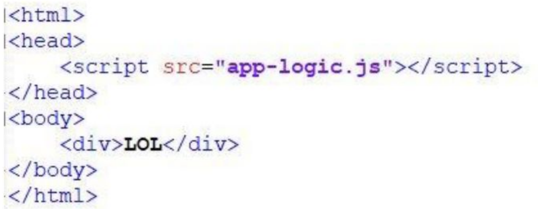


Client Browser
Encryption/ decryption layer
Client Browser
Encryption/ decryption layer
Application server
Web Application
DB
1. Authenticate client-side application code
2. Data Sharing (Privately)
3. Keyword Search

#1 - Authenticate client side code integrity


Client Browser
Encryption/ decryption layer
Client Browser
Encryption/ decryption layer
Application server
Web Application
DB
1. Authenticate client-side application code

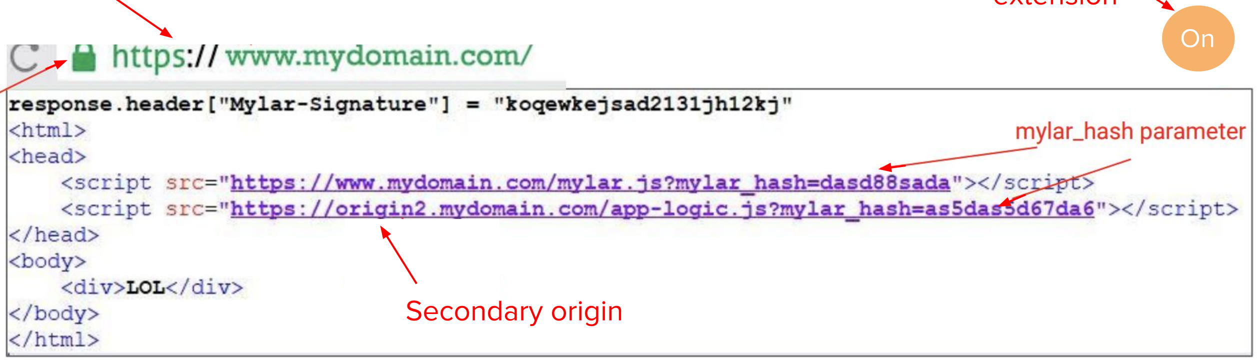
#1 - Authenticate client side code integrity



```
<html>
<head>
    <script src="app-logic.js"></script>
</head>
<body>
    <div>LOL</div>
</body>
</html>
```

Primary origin

Browser extension



```
response.header["Mylar-Signature"] = "koqewkejsad2131jh12kj"
<html>
<head>
    <script src="https://www.mydomain.com/mylar.js?mylar_hash=dasd88sada"></script>
    <script src="https://origin2.mydomain.com/app-logic.js?mylar_hash=as5das5d67da6"></script>
</head>
<body>
    <div>LOL</div>
</body>
</html>
```

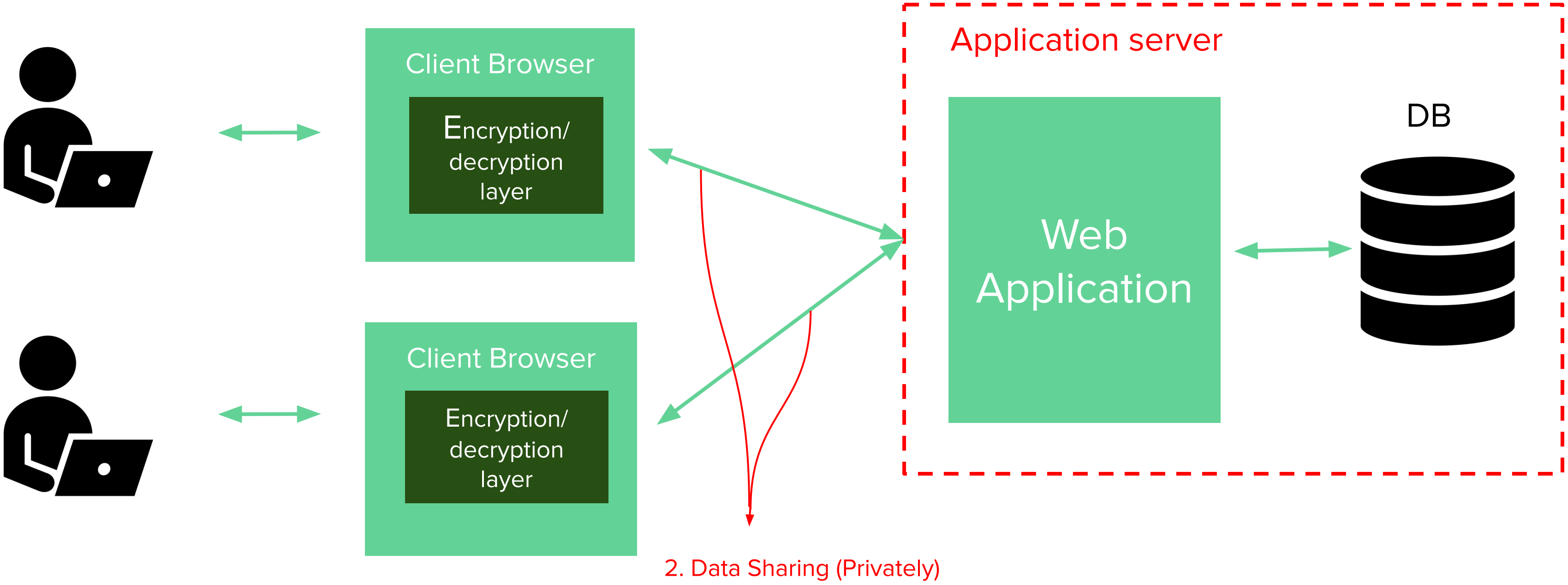
X.509 Certificate with mylar_pubkey

mylar_hash parameter

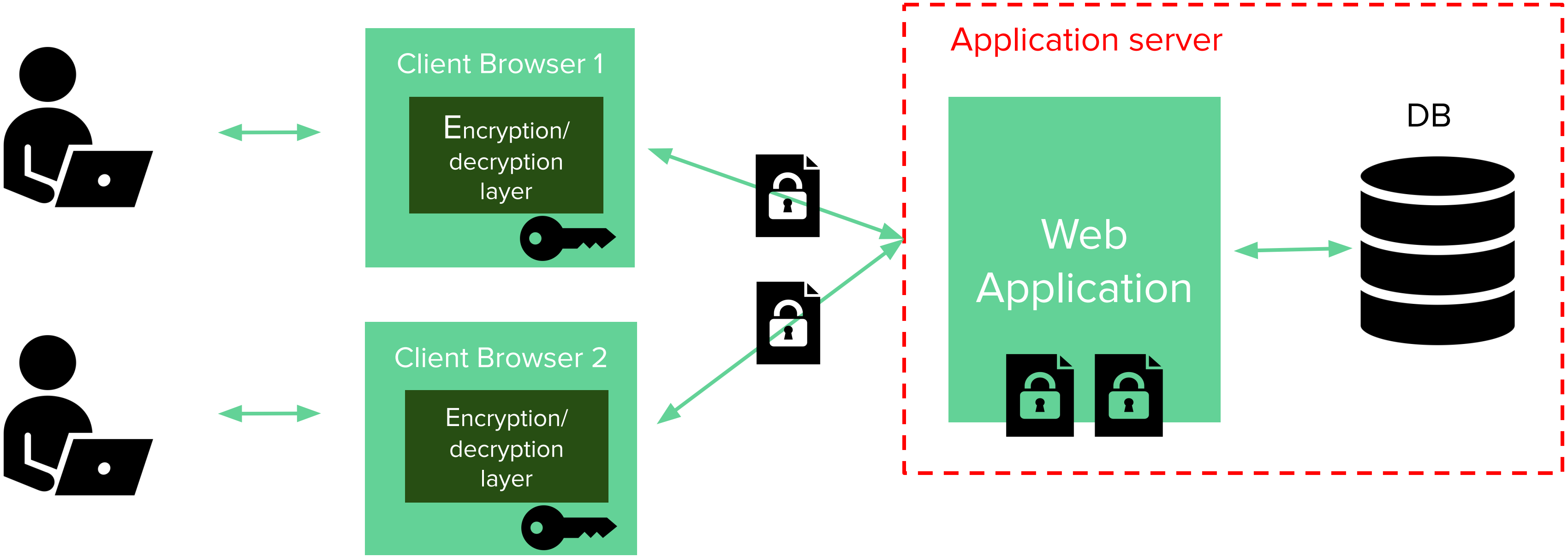
Secondary origin

Source:https://www.google.com/search?q=mylar+ensuring+code+integrity&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjcilH449TsAhUhneAKHT0DAiIQ_AUoAnoECAwQBA&biw=1527&bih=883#imgrc=2j1XzKe_ydvilM

#2 - Data Sharing


Client Browser
Encryption/ decryption layer
Client Browser
Encryption/ decryption layer
Application server
Web Application
DB
2. Data Sharing (Privately)

#2 - Data Sharing



Client Browser 1

Encryption/ decryption layer

Client Browser 2

Encryption/ decryption layer

Application server

Web Application

DB

Question

- Alice wants to send messages to Bob privately.
  - How does Mylar’s client create a user?
  - How dies it share document?

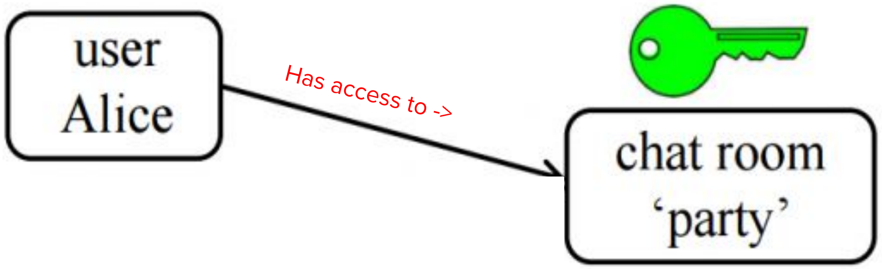
Question

- Alice wants to send messages to bob privately.
  - How does Mylar’s create user?
  - Share document?

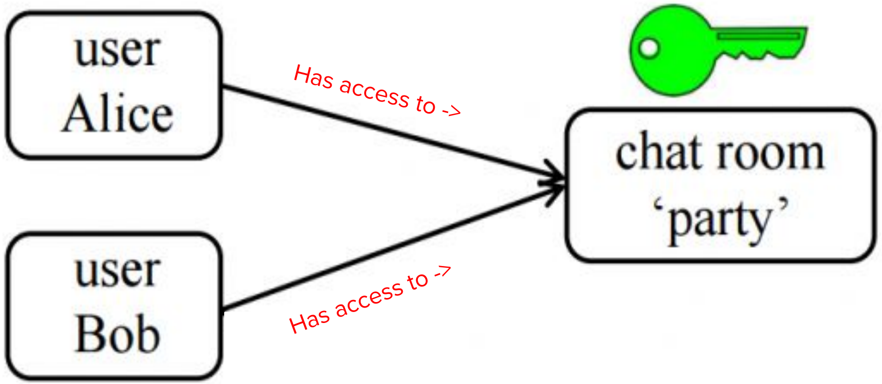
1. creat_user(uname, password, auth_princ)
   - auth_princ can be either static principal or IDP
   - auth_princ helps generate certificate

2. a) Alice generates “Shared Document” pub/priv key pair
   b) Creates wrapped key $E(Priv_{Shared\ Doc}, Pub_{Alice})$

#2 - Data Sharing

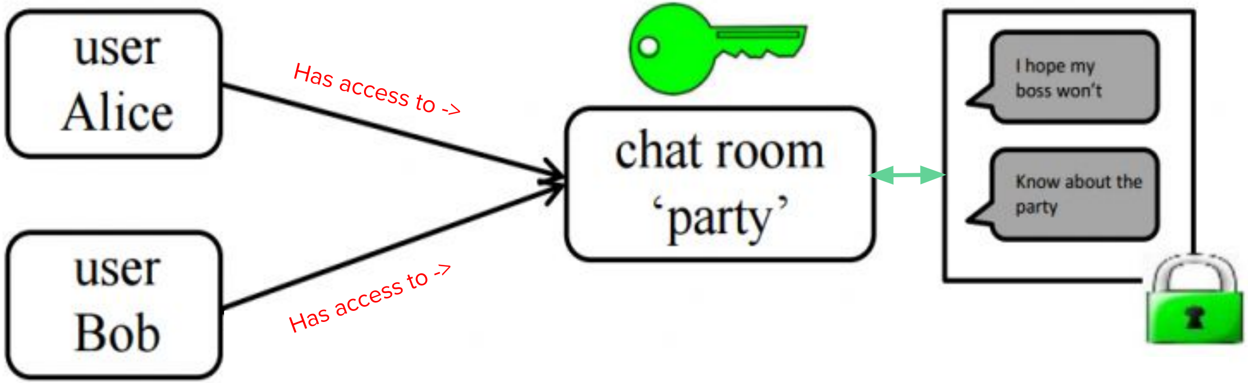


#2 - Data Sharing



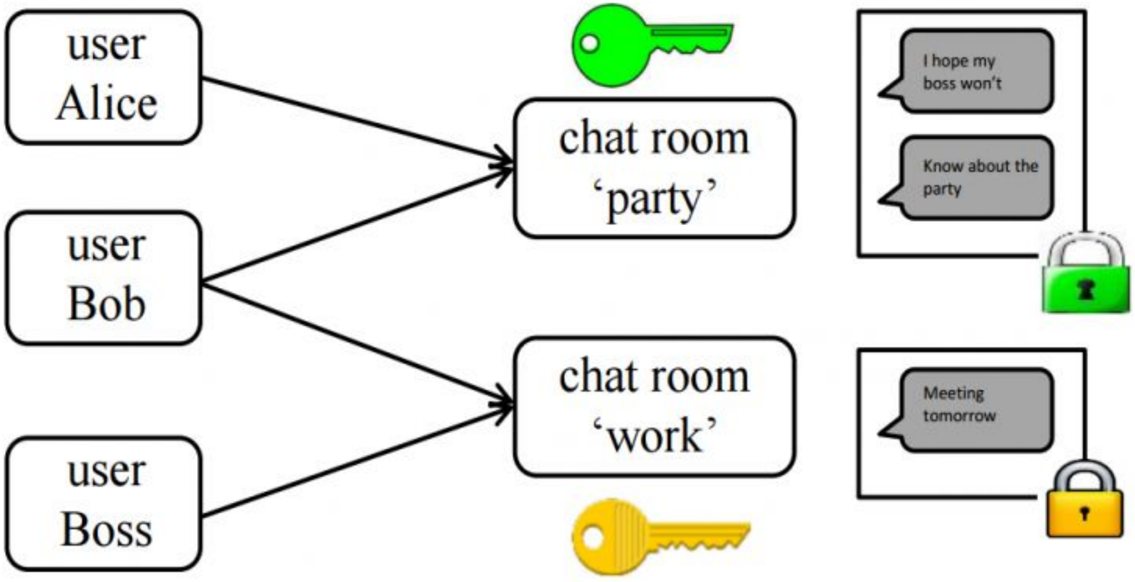
- Bob's principal is granted access to the chat room principal.
- Mylars client uses the public key of Bob to encrypt the document
- Both Alice and Bob have access to the principal for "party"
- Arrows: certificate chains to attest the mapping between principal name and public key

#2 - Data Sharing



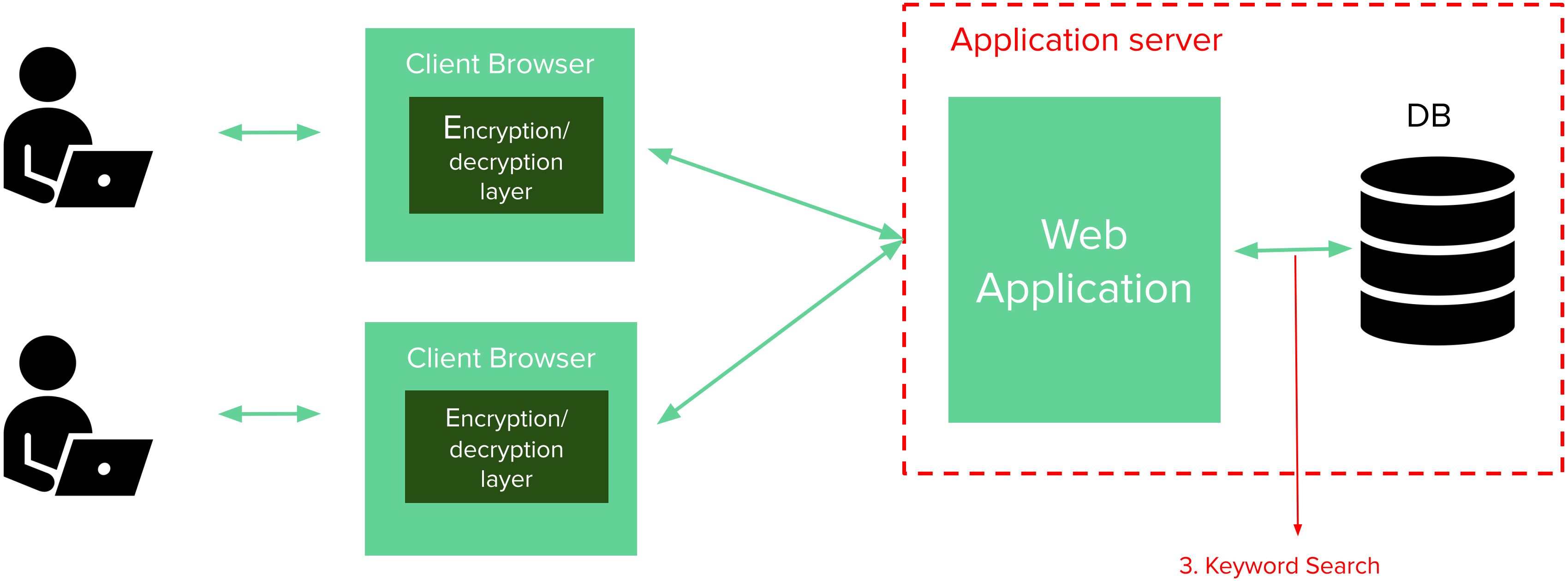
Messages are encrypted with they key for the room’s principal

#2 - Data Sharing - Key chaining



- Private key of ‘party’ is encrypted separately under the public key of Alice and Bob
- The same goes for the ‘work chat’ between Bob and Boss
- These keys are then ‘wrapped’ and stored on the server

#3 - Keyword Search



Question

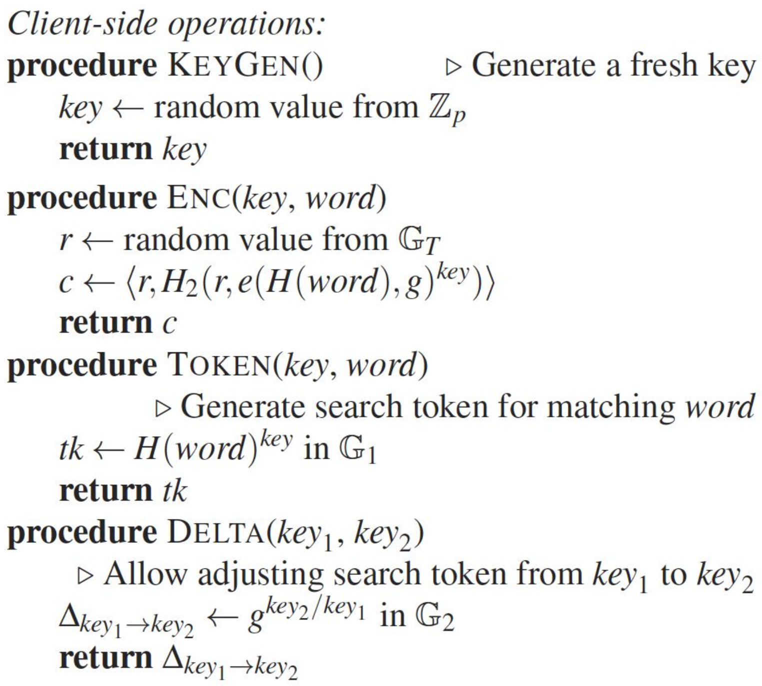
- If a user wants to search for a word in a set of documents on the server, they are each encrypted with a different key. In terms of search, computation over one key at a time has serious limitations.

How does Mylar tackle this?

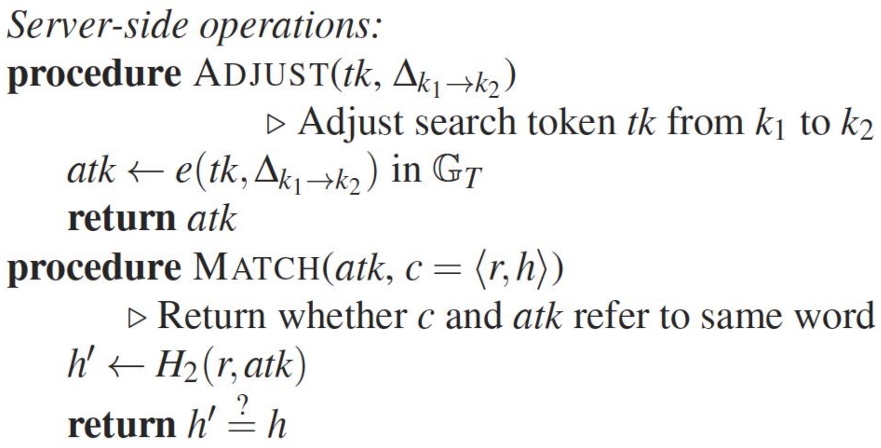
#3 - Keyword Search

- Only need to provide a single search token
  - The server, in turn, returns each encrypted document that contains the user’s keyword, as long as the user has access to that document’s key
- Use delta to adjust one token to another.
- Enable the server to compute token by itself.

#3 - Keyword Search



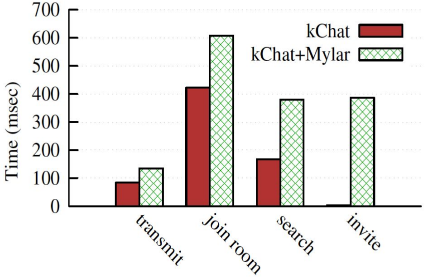
```
Client-side operations:
procedure KeyGen()                              ▷ Generate a fresh key
    $key \leftarrow$ random value from $\mathbb{Z}_p$
    return $key$
procedure Enc($key$, $word$)
    $r \leftarrow$ random value from $\mathbb{G}_T$
    $c \leftarrow \langle r, H_2(r, e(H(word), g)^{key}) \rangle$
    return $c$
procedure Token($key$, $word$)
                        ▷ Generate search token for matching $word$
    $tk \leftarrow H(word)^{key}$ in $\mathbb{G}_1$
    return $tk$
procedure Delta($key_1$, $key_2$)
                ▷ Allow adjusting search token from $key_1$ to $key_2$
    $\Delta_{key_1 \rightarrow key_2} \leftarrow g^{key_2/key_1}$ in $\mathbb{G}_2$
    return $\Delta_{key_1 \rightarrow key_2}$
```

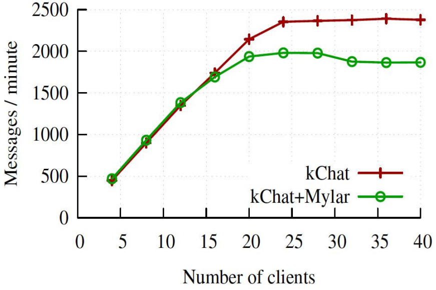


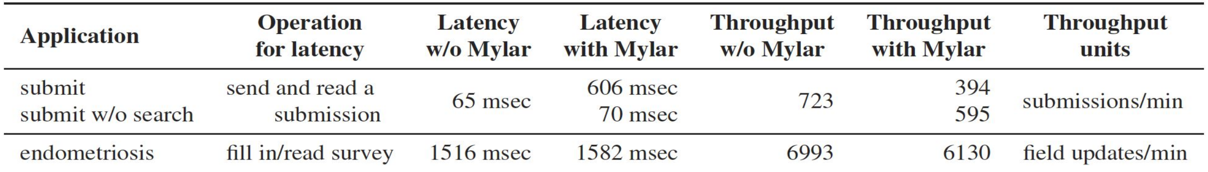
```
Server-side operations:
procedure Adjust($tk$, $\Delta_{k_1 \rightarrow k_2}$)
                        ▷ Adjust search token $tk$ from $k_1$ to $k_2$
    $atk \leftarrow e(tk, \Delta_{k_1 \rightarrow k_2})$ in $\mathbb{G}_T$
    return $atk$
procedure Match($atk$, $c = \langle r, h \rangle$)
                ▷ Return whether $c$ and $atk$ refer to same word
    $h' \leftarrow H_2(r, atk)$
    return $h' \overset{?}{=} h$
```

Limitations

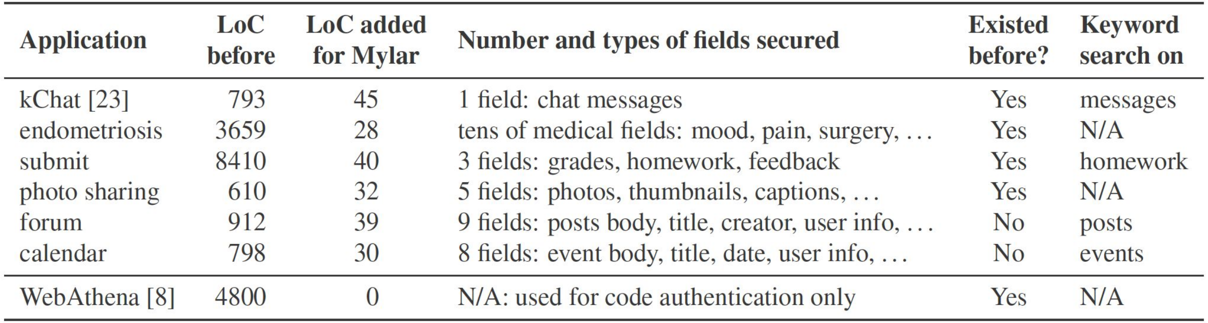
- 4x Space Overhead for kChat
  - ○ Principal graphs (storing certificates and wrapped keys),
  - ○ Symmetric key encryption
  - ○ Searchable encryption







Application	Operation for latency	Latency w/o Mylar	Latency with Mylar	Throughput w/o Mylar	Throughput with Mylar	Throughput units
submit submit w/o search	send and read a submission	65 msec	606 msec 70 msec	723	394 595	submissions/min
endometriosis	fill in/read survey	1516 msec	1582 msec	6993	6130	field updates/min



Application	LoC before	LoC added for Mylar	Number and types of fields secured	Existed before?	Keyword search on
kChat [23]	793	45	1 field: chat messages	Yes	messages
endometriosis	3659	28	tens of medical fields: mood, pain, surgery, . . .	Yes	N/A
submit	8410	40	3 fields: grades, homework, feedback	Yes	homework
photo sharing	610	32	5 fields: photos, thumbnails, captions, . . .	Yes	N/A
forum	912	39	9 fields: posts body, title, creator, user info, . . .	No	posts
calendar	798	30	8 fields: event body, title, date, user info, . . .	No	events
WebAthena [8]	4800	0	N/A: used for code authentication only	Yes	N/A

Conclusion

Mylar supports

- Keywords search over documents encrypted with different keys
- In the presence of an active adversary, share keys and encrypted data safely
- Verify Client-side application code
- Few changes to an application, and modest performance overheads
- Cannot guarantee data freshness, or correctness of query results.

Discussion

- Thoughts?
- What are some challenges to this model?
- Is this model applicable to large scale applications?
- How is Mylar different from CryptDB?