Client-side encryption for files

to implement a zero knowledge system

Google Summer of Code 2017 **Drupal** Project Proposal

Personal Information

Full Name : Tameesh Biswas
University: Indian Institute of Technology, Patna (IIT P)
Graduation Date : May 2020
Major : Computer Science and Engineering
Location: India
Timezone: UTC + 5:30
d.o. Profile : https://www.drupal.org/u/tameeshb
IRC : tameeshb
Preferred time of day for virtual/video interview: UTC 1700 hrs
Github : https://github.com/tameeshb

Contact email: tameeshb@gmail.com
tameeshbiswas1998@gmail.com
biswas.cs16@iitp.ac.in

Project Information

Which project idea sparks your interest and why?

"Client-side encryption for files" is the project that i'm the most interested in, as it is related to the field of web-security and cryptography, two of the topics in computer science that I find the most interesting. Also, the concept of zero-knowledge systems amazed me and because I love taking challenging tasks, this is a perfect fit for me and I would love to and am excited to work on this project.

I've already built a Drupal module that demonstrates the basic idea of Public key cryptography on the client side.
(https://www.drupal.org/sandbox/tameeshb/client_side_crypto)

Also, I've been working on another module implementing encryption and storing files on the client side.

In a confidential post/article sharing sensitive information, the embedded illustrative images and attached files might contain more detailed information than the plain-text article that only provide an overview to the actual in-depth data in the document. Hence, having only the fields encrypted is of no use if the images and other files related to that secret article are leaked by the attackers, possibly posing a higher threat than leak of regular article field text.

Also, the implication of zero knowledge systems is very essential to the user so that they can rely on the CMS so that their data is not being stolen, and they are not being spied upon. This way, sensitive data is already being encrypted before being transmitted.

There are reports of several incidents where the organisation, both private and government, hosting the user's data are reportedly spying on them by tapping on the user's data.

Project on Ideas page : https://groups.drupal.org/node/515848#project1

Project issue page : https://www.drupal.org/node/2629962

Objective:

To build a complete module for Drupal 8 site to make it zero-knowledge so that the users can rely on the site when uploading sensitive files, including images on secret posts, not having to worry about data being stolen in case the server gets compromised.

Detailed Explanation:

A brief on some relevant encryption standards/cryptosystems in use:

- Zero Knowledge Systems : Zero-Knowledge applications offer meaningful privacy assurance to end users because the servers running the application cannot read the data created and stored by the application.
  https://security.stackexchange.com/questions/66323/how-could-a-system-be-zero-knowledge
- Types of encryption being used are:
  - ***Public Key cryptography:*** This method produces an asymmetrical pair of keys(a public key and a private key) where the public key can be used to only encrypt the data and unlike symmetric cryptography, the same key can't decrypt the ciphertext. For decrypting the ciphertext, we will need the private key. There are several Public-Key cryptosystems:
    - **RSA** was one of the first asymmetric key cryptosystems.
    - **Elliptic Curve Cryptography** is a technique to create faster, smaller, and more efficient cryptographic keys and is used by most major modern systems today.
    - **ECDH:** elliptic curve Diffie–Hellman algorithm for key exchange using elliptic curve cryptography.

    https://en.wikipedia.org/wiki/RSA_(cryptosystem)
    https://en.wikipedia.org/wiki/Elliptic_curve_cryptography
  - ***AES Encryption:*** This method of encryption is fairly reliable block cipher and is a standard for symmetric key encryption.
    https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

What we should use:

After some considerable amount of research and discussion with my mentor and others on zero-knowledge systems and related topics in cryptography, I've compiled this section proposing what cryptography standards/cryptosystems must be used also considering what most major modern applications today use:

- AES-256: This cryptosystem shall be used for encrypting the files being uploaded by the user as it is a secure standard for symmetric-key encryption.
  AES is a block cipher and there are several modes/algorithms of implementing it Among all those cipher modes, we would use the **CBC/CTR mode**. It is also the most used mode for other modern applications using AES for data encryption. Whatsapp security whitepaper and Wire Messenger Security Whitepaper
- Public-Key encryption: For the purpose of our project, we will be using asymmetric-key encryption for encrypting the group-access-keys for each user to generate their "access-keys".
  There are several cryptosystems implementing public-key cryptography, one of the first and most used ones was the RSA standard. For the initial phase of development, we will implement RSA for the asymmetric keys and later on extending to a rather modern cryptosystem, called *elliptic-curve cryptography*. EC will be implemented and made the default for asymmetric encryption in the module.
- Further specifics in *elliptic-curve crypto:* In EC, there are *several curve types* that are used to generate the key-pairs, among all of them, ***Curve25519*** will be used for both security and efficiency reasons. Also, this is the algorithm being used in most modern crypto applications: Whatsapp security whitepaper and Wire Messenger Security Whitepaper
- It is one of the fastest ECC curves. it is not covered by any known patents, and it is less susceptible to weak random-number generators.[wikipedia reference]
- For the initial development phase 2048-bit/4096-bit keys can used for the RSA encryption, however, later studying out different factors like security and speed, we can choose a better default and provide configuration based on requirements of the system admin. More info: Javamex - rsa_key_length
- We will **avoid** using BLOB datatype to store in the database and instead use the regular method of having the data on the server's storage drives. Using BLOB can depreciate Database Querying performance.
- The primary encryption is handled on the Front end, on the browser using the JS libraries. The private key, used to encrypt the group-access-key will be stored on the client side and not on the server to make this a "zero knowledge system".
- More discussion on this idea has been done at
  https://www.drupal.org/node/2629962
- About the HTML Web Storage API (to store the private key of the user)
  https://www.w3.org/TR/webstorage/

Problems with the Existing System (Server side encryption with File encrypt):

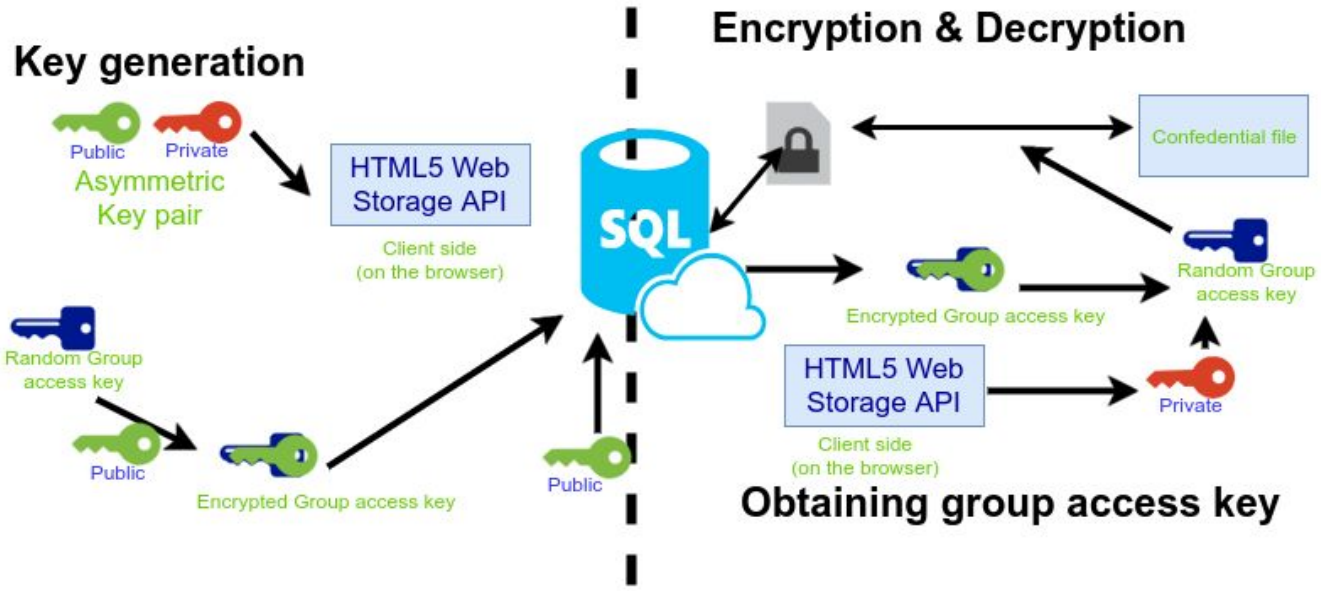
- Currently, whenever the data is sent over to the server for encryption, the encryption keys are stored on the server side, making it vulnerable in case of a breach into the server as the attacker would have access to the keys.
- We are considering a case where the server is unreliable and that it could be compromised some time in the future and hence any data sent to the server for encryption can be read by the attackers even before it being encrypted.

Proposed Solution:

- All the above problems can be solved by encrypting files, first, using a symmetric key and then encrypting the symmetric key for every user using their public keys, all of which will take place on the front end and ONLY encrypted information(Public key + encrypted access keys + ciphertext) is passed to storage provider.
- The generated private keys will be stored on the user's browser, preferably using "HTML Web Storage API" (https://www.w3.org/TR/webstorage/)
- The private key when generated will be exported to the user's device to store it safely and later move it to another device to gain access to content. There would be an interface to load the keys from .pem files to the browser storage.
- Encrypting whole data using the user's public key might be a problem,as then we will have to have multiple copies of the same data and also because, when the user changes their passwords, all the data would be needed to be decrypted again and then re-encrypted with the new keys. Also, that would create multiple copies of the same data encrypted by different keys that will take up a lot a space and computation.
- To address the above problem, we encrypt keys rather than the data.

We could be dealing with two types of data here,
1] The first one is where the data is intended to be shared with only a fixed group of users and no-one else should be able to access this private content.
(Eg. Private posts to a group audience)

- The following points explain how the mechanism would work.
- It is necessary to generate a set of keys to later encrypt or decrypt data for every user in the group:
  1. **Generate a Public/Private key pair** for each user when they log in initially.
  2. **Store** the private key **locally** using **HTML5 web storage API.**
  3. **Generate a random group-key** that will be used to encrypt all private data shared in that group. **This key will NOT be stored on the server.**
  4. The module should then generate **access-keys** by encrypting this group-key using each user's public key.
  5. Then the following are sent over to the server and stored there:
     - Unencrypted Public key for each user.
     - An Access Key for each user.



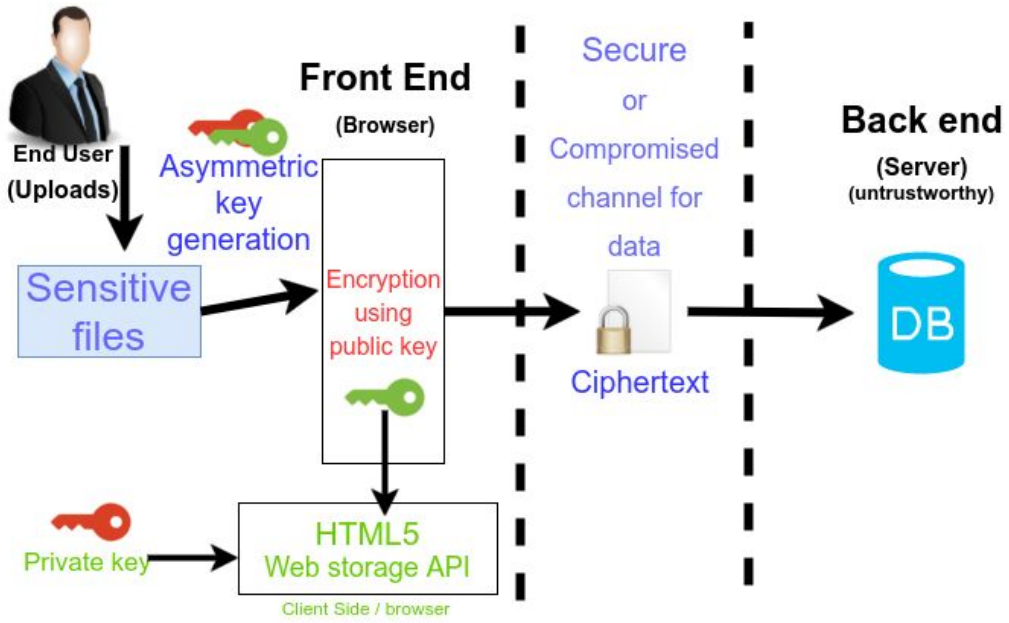
(Flowchart key-generation and encryption/decryption. Created by me using draw.io)

- Now, if a user wants to access some post that is encrypted:
    - The browser will fetch the author's (Encrypted) access key
    - The data to be displayed is currently encrypted using group access-key(This key is encrypted using the user's public key).
    - The private key stored on the browser will be used to decrypt the group-key using Public Key encryption.
    - This obtained group key will then be used to decipher the private content.

- Then, If a member of the group wants to post some private data:
    - Encrypted access key is fetched from the server to the client side.
    - This key is decrypted using locally stored user's private key.
    - This key is used to encrypt the data and only the ciphertext is sent to the server.
- When there's a new user in the group, the group-access-key will be decrypted and then another user-access key will be generated by encrypting it using their public key.

2]The other type of data we might be dealing with could be data that only the author can see. (Eg: Storing credit card data or any other data that they don't want to share with others)

- A possible way of solving the above problem would be to
    - Generate asymmetric key pair.
    - Encrypting the user's sensitive data on the frontend using the public key.
    - Storing the private key locally on the browser using HTML5 Web Storage API
- In this case, we won't need to worry about same data being duplicated as this data is meant for only a single user.
- Alternatively,for this, a new group can be created containing the user alone can be created.



(Flowchart created by me using draw.io)

Possible Challenges

- Also, if there is a new user, the user would have to wait until another user of the same group comes online for them to provide the new user with their new access key encrypted using the new user's newly generated public key.
- The problem with storing the keys locally is that the user won't be able to login using a different device.
- Also, if the user clears their browser data, the browser doesn't have the user's private key anymore, so a mechanism would be needed to reload the key into the browser from a ***.pem file***, this will solve the previous problem too.
- There would be a loophole in implementing this where the attacker can replace a user's public key by his own public key and would get a access key using that public key hence gaining access to the data.
  - A way to prevent this would be generating a public key certificate(wiki) to prove the ownership of a public key.
  - Also, a password based hash of the public key can be created and stored on the server for the user and the authenticity of the public key can be verified. Here, the user's password is being used for checking authenticity of key.
  - This issue needs further discussion to come up with more solutions and settle with one.
- To serve our purpose, there is a plethora of open-source crypto libraries available on the internet. I found out the following libraries could be used as they have fairly stable releases as listed below.

External libraries/APIs with alternatives that can be utilised:

- openPGP.js - https://github.com/openpgpjs/openpgpjs for RSA/EC or
- Jsencrypt - https://github.com/travist/jsencrypt for RSA/EC

- PKIjs - https://github.com/PeculiarVentures/PKI.js for encrypting files
- JSAES - http://point-at-infinity.org/jsaes/ for AES or
- slowAES - http://code.google.com/p/slowaes for AES or
- Cryptico - https://github.com/wwwtyro/cryptico for both RSA/EC and AES
- HTML Web Crypto API - https://www.w3.org/TR/WebCryptoAPI
- HTML Web Storage API - https://www.w3.org/TR/webstorage

Mentors for the project:

- Colan Schwartz (colan) Enterprise Cloud Architect,Colan Schwartz Consulting https://www.drupal.org/u/colan
- Adam Bergstein (nerdstein) Associate Director of Engineering,CivicActions https://www.drupal.org/u/nerdstein

I'm presently in contact with the mentors and am taking frequent inputs from Colan on my proposal and am discussing the ideas.

Expected Deliverables:

- Proper architectural document of the module before beginning to code.(by week 1)
- RSA key generation and local web storage with private key management(store and load .pem).(By week 3)
- Interface for creating, adding and deleting users from groups/roles.(By week 1)
- Basic core functionality for uploading, encrypting and storing the data.(By week 6)
- Add support and make default Elliptic Curve Cryptography and other flexibilities.(By week 10)
- Writing unit tests and documentaion after every part built.(Simultaneously/Adjacent with every component built)
- Proper documentation for the entire module covering all it's components.(By week 12)
- Video demonstration for using the module.(By week 12)

Timeline for the project:

Community Bonding Period:
May 4 - May 30

- Start with discussing the architecture of the module with the mentors.
- I plan to complete with the "Examples for Developers" Module to better understand the core APIs being used in Drupal, read all the best practices used to develop for Drupal. This would facilitate me in making a better Module.
- Also, i'm planning to read all relevant documentation for the related Core and external APIs in use and test experiment with the function of the APIs and the modules. This shouldn't take a lot of time as I'm familiar with building Drupal modules from scratch and coding a similar module.
- Test all the js APIs and choose the better ones to use with discussion.

Week 1 :

May 31 - June 6

- **Architecture doc, Module skeleton, js libraries, DB schema initialisation, Group creation and adding users:**
- Once the architecture is agreed upon fully, it needs to be documented before starting to code to have all ideas clear, like how the module would temporarily store all the keys and send the encrypted ciphertext.
- Completing the module skeleton including the `.info.yml`, `.routing.yml`, `.install`(for the database table creation).
- Creating the .libraries.yml for creating the libraries to include the js assets.
- Creating the tables in the database using `.install`, implementing `hook_schema()`.
- Building the initial admin interface.
- Writing php scripts to create the groups/Drupal roles and and store in the database.
- Adding new users to these groups.
- Conclude the week with writing the tests and documentation for all the sections built so far.
- Iterating over work done based on mentor reviews and phpUnit results.

Week 2 & 3

June 7 - June 20

- **Asymmetric key-pair management and restoring interface:**
- We will be utilising the Entity API/Keys module to add fields for storing the user access keys and the public keys associated with each user.
- On login: Check if the user has been added to a new group. If yes, then:
- Modifying default login behaviour for the user using `hook_user_login` to redirect the user to a page that would generate their public/private key pair and store the private key to browser using `localStorage.setItem()` (Web Storage API) and let them download a copy to their local machine, preferably in a .pem format.
- When the module is installed we must force logout all the users for the effect of `hook_user_login.`
- The above page would also display a warning saying that the key should be kept safely for access to data.
- Build an interface to let the user load/reload the private key if emptied browser cache or to gain access from a new device.
- Managing uploading the public key using an AJAX request from that page to the php script which will use `Hook_user_insert` to add the public key to the respective user's `user entity field.`
- Alternatively, the keys could be stored using the Keys Module. This might be favourable as the module would take care about the security measures of storing Keys. Either way, the keys that we are uploading to the server, alone won't be sufficient to decipher any information.

- The “user-access key” for that user isn’t generated yet, we will need to mark it in the database specifying that user-key for this user is yet to be generated.

- ***Generating and storing group access keys:***
- Generating symmetric keys (group-access-keys) for file encryption, one per group, at the admin’s client side when the admin creates a new group.
- Temporarily, storing the keys on the front end, possibly in a hidden form field or a js variable.
- Retrieving Admin’s Public key from the `user entity field/Keys Module` and encrypting the AES-256 bit key generated in the beginning of the week on the frontend to generate first user-access-key.
- This access key is later sent through an AJAX post to store on the backend for later use while accessing data.
- Adding features to the admin configuration panel for what fields to encrypt.
- **Writing tests** for all the work done in week 2 & 3 parallely.
- Iterating over work done based on mentor reviews and phpUnit results .

- I will be away from home for a while during week 2. I’ve discussed with my mentor about it and he is fine with it as long as everyone is informed about it. I’ve accordingly planned the timeline to be able to complete the work from week 2 in week 3, just in case of any delay. Also, if some work is left, I will surely make up for it in the weekends and extra hours on weekdays.

Week 4:

June 21 - June 27 (Phase 1 evaluations)

- All the work done so far will be evaluated by the mentor to review/evaluate the progress and performance for the first evaluation will be sent to Google using the GSoC web-app.
- Documenting the parts of the project built so far in the last three weeks.
- Testing speed of encryption and decryption for different key sizes for RSA key sizes.
- Suitably, providing a default value after several rigorous tests and proper analysis.
- **User-access keys**
- When admin or an existing user logs in, default login hook can be slightly modified to check if there are any pending users who have uploaded public keys but are awaiting their user-access-keys. If there are any such cases:
- Same method to retrieve Keys in the previous week can be reused to obtain the public key of the new user and the user-access-key of the existing user.
- With the above two provided by the server and the private key from Web-Storage API, the group-access-key will be re-encrypted using the new-user’s public key after checking authenticity of the public key.
- At this stage, we will have completed with the key-manager,the part with dealing with all the key generation and storage. The part left for an alpha version of the module is the file encryption and decryption parts.

Week 5 & 6:
June 28 - July 11

- **Overriding default upload actions, encrypting and uploading the file.**
- Retrieving the user-access key using the same method used in the previous weeks.
- Decrypting the user-access key using the locally stored Private key.
- Next part to take care of would be overriding the default action on adding a new file for a basic page/article. event.preventDefault() shall be used to not let HTML submit the form in the traditional method.
- Instead, the file would be encrypted using JS library for AES.
- For now, the file name is being kept as-is. It can be encrypted and managed in a later stage after having a working alpha/beta.
- The file is uploaded to the server using the regular configuration for uploading files after converting to ciphertext.
- The rest of the article/Basic page is sent as-is(until optional Field encryption is used).
- Writing tests for file encryption and file upload with documentation.

Week 7:
July 12 - July 18

- **Overriding default article viewing hooks, retrieving and decrypting the file.**
- Retrieving the user-access key using the same method used in the previous weeks.
- Decrypting the user-access key using the locally stored Private key.
- The user could be requesting for the encrypted files in two ways, either an embeded image or a file to download.
- Default PHP renderers can be extended to return ciphertext with flags saying that this is encrypted data and rather fill the embedded image/attached file using JS after fetching and decrypting the ciphertext on the frontend.

Week 8:
July 19 - July 25 (Phase 2 evaluations)

- All the work done so far will be reviewed and sent for the first of three evaluations.
- Writing documentation for parts of the module built in the last 4 weeks.
- Writing unit tests in for functionalities built in the previous week.
- At this point, we should be ready to release a beta of the module and move it to drupal.org as a project and get the community to use it and provide feedback on it.
- Now methods for removing users from groups can be created. The database entry for their user-access-key will be deleted.
- Start integrating Elliptic Curve as an alternative to RSA.

Week 9 & 10:
July 26 - August 8

- These weeks would be used to add **optional additional features and offer customizability** for the module to meet the system admin's requirements.
- **Key length** for RSA:
  "With every doubling of the RSA key length, decryption is 6-7 times times slower." - online article .
  Hence, we must choose a suitable default RSA key length considering the fact that doubling the key size slows down encryption and decryption, as done in week 4. I will further add options for an advanced admin to be able to choose a key-length depending on their requirements.
- Work on integrating **Elliptic Curve** cryptosystem was started in week 8, upon which the work will be completed and made default after discussions. As mentioned before, EC is the future and already being used by most modern applications for asymmetric encryption.
- Optionally, the name of the file can also be encrypted and a reference can be stored on a database.
- Dealing with recommended changes for best performance like cache on the front-end and back-end.
- If it is considered to be more convenient for the users and secure, the module can be extended to store multiple keys for different devices for the same user so that they won't need to maintain a .pem key which gets cumbersome for an end-user.
- Re-encrypting access key if private key is lost ensuring no backdoors are opened.

Week 11:
August 9 - August 15

- Writing tests and documentation for all new features added in the last two weeks.
- Improving and finalising the documentation.
- Optionally extending the module to encrypt fields using the same keys.
- Tests specifically targeted at testing the security standards of the security features used in the module.

Week 12:
August 16 - August 21

- Write final tests and fix bugs and other issues.
- Creating Video documentation for a tutorial example to use the module. Apart from the text documentation, I will make a screen recording to demonstrate a use case from the end-user's perspective of the finished module.

Final Wrap-up
August 21 - August 29 : Final Week, submitting final work.

Q&A

Which aspect of the project idea do you see as the most difficult?

The part of the project where the default file submit action is being overridden to implement encryption of the file and then send the file instead of the regular unencrypted file to store on the backend would be a challenging as there are several variables to figure out like where the file would be stored before sending it to the backend and other concerns about handling the file on the front end. Also, as this is a project providing security features, so every step taken to build this module must be carefully evaluated to have no backdoors / loopholes, this needs to be taken special care of.

Which aspect of the project idea do you see as the easiest?

According to me, the easiest part of the project should be to store the keys on the backend in the database as I will primarily be using already present modules for this purpose.

Which portion of the project idea will you start with?

I will start off with writing an architectural documentation for the module and then building the module skeleton.

How will you deal with project, task, and time management? Will you utilize software? If so, which tool and why?

For project management:

As discussed with colan, we will use github (d8-contrib-modules) for the initial stages of development and later move it to a drupal.org project. That way, once the development release/beta is ready, we can roll it out on drupal.org for the community to use and provide feedback on it.

For task management:

I will make a separate issue for every task necessary in execution of the project, as recommended by my mentor. This will make it possible for my mentors to track my progress on the project every week and keep a check on whether the proposed project timeline is being followed apart from regular meetings with the mentors. There will also be a detailed blog post every week on blog.tameesh.in detailing on what has been accomplished in that week.

For time management

Most of the GSoC period is during the summer vacations at my college.
I will strictly adhere to a fixed time schedule.
If I'm selected for GSoC '17, I will work between IST 3 pm to 6 pm and 10 pm to 4 am on week-days totaling to 8+ hours a day = 8 * 5 = 40+ hours a week.

Also, after college reopens on 26th July (week 9), as most of the work will already be done and the beta released I will work for 6 hours during weekdays for the rest of the work left and more if required, on weekends.
In case of any delays, I will put in more time whenever required.
I will be available on IRC during my working hours.

How often will you communicate with mentor? How will you communicate with mentors?

As my mentors have suggested, we would have one or more meetings in a week. We would meet more frequently towards the beginning to clearly lay out the direction and architecture of the project and lesser meetings as I get started coding once we have the direction figured out, to keep a check on the progress.
Tools that would be used for communication would be using ***Wire,*** an *open-source* (only client-side source) client-side encrypted communication service.
Using an open source tool for developing an open source project promotes the spirit of open-source software. Also, this is demonstrates how we can use open source tools to build great software!

Project Difficulty: Difficult

Future Work (to be done after completion of this project)

- Keeping an eye on the latest cryptography standards and integrating the newer and better cryptosystems as time passes and purging out some standards as they become obsolete.
- I will actively maintain this module and be active in the issue queues to provide help for module-related issues.
- There can be multiple options that can be provided to the site admin for support of variables like different key sizes and different algorithms for the same encryption standard.
- Any drawbacks / any suggestions for ways in which the security of the module can be made better will be actively worked upon by me.
- Merging key management for client side field and file encryption so that users don't have to deal with two different keys.

Drupal Core contributions so far:

- I have been actively participating in the Drupal community since several months.
- I have contributed many patches for Drupal core and other contributed modules.
- I have reviewed several patches and manually tested some.
- Presently, I have ***11 patches committed*** to Drupal ***core***.
- I've also helped a few relatively newer contributors get aboard the drupal community and in the issue queues.
- Also, I have ported two modules d7 to d8 from contribkanban.

Why Me?

- I've built a sandbox module for drupal to demonstrate the implementation of Public key cryptography on the client side, which covers a part of the idea of this project.
  https://www.drupal.org/sandbox/tameeshb/client_side_crypto
- I've also been working on a sandbox module for front-end file encryption.
- I have a keen interest in the field of web-security, cryptography and ethical hacking and have a background in fundamentals of all the three.
- I have worked on and have been working on projects implementing some or the other method of modern cryptosystems.
- I have taken courses on Cryptography and have a good understanding in intermediate JS and modern-cryptography which will help me better execute this project idea.
- I am an active member of the Drupal community, and have also helped some newer contributors on IRC / the issue queue.
- I am an active core contributor to Drupal core (11 commits to core) and have utilised the 'Examples for Developers' module to better understand the core APIs in Drupal.
- I have ported two modules from d7 to d8 before giving me a good idea about contributed modules in Drupal.
- I also have a good understanding and hands-on experience of coding Drupal modules from scratch.
- I have also been contributing in issue queues of relevant modules.
- Having done the above, I think I have a decent understanding of how Drupal works and will smoothly be able to execute the plan for the project.

Technical Information

Have you ever worked with Git?

Yes, I've been using git for over half a year now and am quite proficient with it as i've been using it with most of my recent projects. https://github.com/tameeshb

Have you ever utilized IRC?

IRC: tameeshb

Yes, I'm online most of the time that i'm awake, on freenode, #drupal-google and have been communicating with many other members of the drupal community through IRC regularly.

Have you completed Drupal Ladder for GSoC Students?

Yes, I had completed the Drupal ladder for GSoC Students and the Drupal Core Ladders before starting to contribute patches to Drupal. My contribution to following issues show that I create, test and reroll patches that the ladder introduces us to.

- Installing Drupal on VPS: blog.tameesh.in (whois lookup) (blog post link1 link2)
- Patch Reroll: https://www.drupal.org/node/2820347
- Writing patches: https://www.drupal.org/node/2853684
- Using Git: https://github.com/Tameeshb

Have you ever contributed code to Drupal?

I have been actively contributing in Drupal Core and other modules since last year. At present I have 11 patches committed to core and a lot of them in RTBC that are yet to be committed.
https://www.drupal.org/u/tameeshb
https://www.drupal.org/user/3479893/track

Do you plan to continue contributing to the Drupal project after GSoC is finished?

Yes, the Drupal community is the most active open source community that I've got involved in or even known about, the community is full of enthusiastic and expert members who are always there to help any new contributor (or anyone) with any issues they might have. It has been amazing, so far, being active in the drupal community for which I want to continue to contribute to core, other modules, try to become maintainer for several modules and help many more new members to contribute in the following years.

Have you ever built a Drupal site or helped on a Drupal project?

Yes, I've built Drupal sites before and have had some of them online too (blog.tameesh.in). One of the recent ones online was a CMS version of my blog that I had originally coded from scratch.

GSOC Information

Q1. Have you participated in / applied to Google Summer of Code in the past?

No

Q2. Are you applying to any other organizations this year?

No

Q3. What are your other summer plans?

Other than GSoC, I don't have any major plans for the summer except for going out on a short trip in week 2, which I've already discussed with my mentors and they are fine with it as long as everyone is informed about it and the work is on track.

Biography

I'm Tameesh Biswas, tameeshb on IRC, Currently a Computer Science and Engineering undergraduate at the Indian Institute of Technology, Patna (IIT P). I had first used Drupal around 4-5 years ago, after which, since the last couple of months I've been actively contributing to the Drupal open source project, mainly core.
Writing code is what I love to do and what I do most of the day. I had started programming at 12 when I was amazed by the power of computer programming and have been only diving deeper into the world of Computer Science.
I love to build cool and challenging stuff!
http://blog.tameesh.in
https://tameesh.in

Terms and Conditions I agree to attend the weekly check-in meeting and understand that missing two meetings will result in a failure. In case of unavailability, I will positively inform beforehand and try to get a check-in meeting before time.