1. **system assumptions**. The chosen system is a web-based application, as opposed to a native app that would result in a more inaccessible system. The web app was chosen in order to maximize user compatibility, i.e it is supposed to work and be accessible regardless of hardware (phone, computer et cetera) or software (operating systems and such). This ensures that only a web browser and a stable internet connection is required to use the services. The users will be met by a web-interface, i.e a login page (access and user management), communication channels for individual, group and public KTH-wide correspondence, as well as ample security to store and transport said correspondence. The system should be stable and large enough (processing and memory wise) to support a large quantity of concurrent users. The main server should also be stored within the EU, as to ensure that data is not sent to a country where data protection laws are weak, as in, data might be confiscated or stolen with little to no repercussions.

2. **user assumptions.** The users are KTH students, staff, or other individuals somehow affiliated with KTH. These individuals are expected to have a moderate to high technical understanding of how to navigate the web, and are expected to thoroughly learn how to use the web application. As the students divide themselves into many programmes, courses, groups and sub-groups, they expect to be able to create private communication channels, where they will be able to confidentially communicate with their group members. The same goes for private communication with another individual. I assume that they possess elementary cybersecurity common sense like not reusing passwords, using strong passwords, being cautious about phishing attempts and so on.

3. **adversary assumptions.** The adversaries could be both internal and external actors. External actors might want to get access to or disrupt the system in order to extort a ransom. In other words, I consider their motives to be mostly financial, alternatively ideological. I assume that the bulk of Internal threats are made up by students attempting to cheat or disgruntled employees that seek to disrupt the system from within. They could aim to download confidential information by getting hold of stored and encrypted data on the system (or data in transit), which could be achieved by man-in-the-middle attacks or plain social engineering (tricking their fellow students into giving out credentials).

4. **requirements.** The standard set of requirements for data or communication holds, namely confidentiality (that the contents of communication remains safe), integrity (assurance that communication has not been altered), availability (available to all users at all times) and authenticity (that the identities of the communicating parties are verified in each instance of communication). All of these requirements need to be implemented without weighing down the performance of the system too much. Confidentiality can be relaxed based on the scope of the communication (e.g if it is supposed to be public communication). The communication of all students should be encrypted and stored until graduation in case cheating allegations surface, in which case inquiries into communication must be made.

5. **architecture**. The system will be located on a main server, where data and the web-interface will be stored. The user management component also requires that credentials are stored in a hashed and salted format, in such a way that reversing the passwords is extremely difficult. Communication between client web browsers and the web application will be over HTTPS, using TLS (Transport Layer Security), which encrypts all traffic between each and every KTH member client browser and the webpage located on the server. For private communication, a public key infrastructure will be used (PKI), where the public keys of each and every member of the network will be visible to all the other members of the network. The encryption of communication will be client-side, after which it is sent to the main server and routed to the recipient or recipients. I expect the operating system of the main server to be Ubuntu. For group chats, symmetric keys will be used. These symmetric keys will be transported by utilizing the PKI, after which every member of the group will have a key with which they can encrypt communication only readable (or

audible) by the other group members. In order to combat uncovered credentials, two-factor authentication is required upon each login attempt, which can be handled by a third party mobile app.

6. **data at rest**. Data should be stored with the main server, using some database framework (like SQL), where it should also be encrypted, ensuring its confidentiality even if it were to fall into malicious hands. The server stores all the private keys of each and every user. Thus, they can decrypt any stored data if the need arises. All data is encrypted with a different key, depending on who it belongs to. For group specific data and communication, a symmetric key is used for encryption, and the key is transported over the PKI to each and every member of the group.

7. **data in transit**. Messages directed to various recipients are sent from client to server over TLS, as well as credentials for the user management system. These are all encrypted with the public keys of their recipients, along with a signed hash of the message to ensure integrity. The hash is signed with the private key of the sender. The receiver verifies the hash with the public key of the sender, thus assuring themselves that the message has in fact been sent by the sender, has not been tampered with, and has its integrity intact. The encrypted message and signed hash is sent to the main server, which stores and reroutes the message to the appropriate recipient or recipients.

8. **cryptographic algorithms and protocols**. My algorithm of choice is RSA, a PKI. Without delving too much into technicalities, this works by generating a key-pair, one private and one public. These are generated by selecting two prime numbers, preferably by a pseudo-random number generator (PRNG) which uses a secure and reliable seed generator as input. The public key is distributed and can be used by anyone to encrypt a message, by way of modular operations that are unfeasible to reverse. For communication in groups, the AES-256 in OCB mode algorithm is used, which is a commonly accepted symmetric encryption standard. AES works by encrypting blocks of data using the aforementioned symmetric key. Each block is encrypted 14 times, with 14 different keys, all derived from the original key. OCB mode ensures that the encryption is authenticated, i.e that the integrity of the message is retained and the sender authenticated. This mode therefore protects against certain types of attacks, like replay attacks (intercepted encrypted messages that are retransmitted), eavesdropping and message forgeries.

9. **implementation**. Certificates (and keys), algorithms and basic cryptographic functions and utility for both TLS and RSA can be implemented with the popular public cryptographic library OpenSSL. If the main server uses Ubuntu as an OS, Open SSL comes pre-installed, meaning that its functions can be used with simple function calls from any written code or architecture. Due to its popularity and age, OpenSSL is supported by many systems and languages, making it ideal for use when only using basic algorithms such as TLS and RSA. Other alternatives to OpenSSL are wolfcrypt and BouncyCastle, but using these instead of the old faithful OpenSSL is unnecessary, as they provide identical services. For two-factor authentication, Google authenticator is the most common and convenient tool.

10. **parameters**. The largest commonly occurring key length in the RSA algorithm is 4096 bits, which is what should be used in this system. This key-size is large enough to practically ensure that it cannot be brute-force guessed or reversed using any conventional means of figuring out the correct key. When group chats are utilized, a symmetric AES-256 key is transported by way of RSA. This size, a prime number of 256 bits, is the highest supported bit size by the AES algorithm, and is practically unbreakable by brute force with regards to available modern computing power.

11. **ensuring randomness**. The keys for RSA 4096 are created with two two 2,048-bit prime numbers. These initial prime numbers are generated by way of a PRNG, which relies on a seed from a secure source to determine the resulting sequence of bits. This seed can be provided by measuring ever changing cosmic

radiation for instance, or some other hardware solution that can be considered random. If performance is desired, a software-based random number generator (RNG) can be used to create a random seed. These solutions however are prone to not being as random as physical phenomena. The AES key is generated in an identical manner to the RSA keys.

12. **trade-offs and risks**. AES OCB mode, as opposed to other AES modes, introduces little computational overhead when ensuring the integrity of each and every encrypted message. Other alternatives like GCM mode offer similar functionality but are very slow. Thus, the chosen mode for group communication offers the best tradeoff between performance and security. OCB is however used in conjunction with a 256 bit key size. This larger key size makes the key much more difficult to brute-force, it is however also slower than its smaller key size sibling, AES 128, as it requires more encryption rounds (14 vs 10). The same tradeoff applies to RSA 4096 over RSA 2048, as in: a larger key size that is much more difficult to brute-force guess, but also more costly and slower to encrypt with. As we want the most secure system however, we opt for the longer algorithmic key lengths.

13. **legal considerations**. According to GDPR, an organization, in this case KTH, can store their members confidential information (students, faculty members and so on) if they (1) have a legitimate reason to do so, which they have (anti-cheating measure), (2) process the data in a sufficiently safe manner, the data in this case being protected by encryption, and (3) That they do not transfer the data to a country outside the EU that does not have adequate data protection laws. As long as these three points are followed, no legal trouble should ensue.

14. **usability considerations**. As one might expect, the system should be usable by all members of KTH. This includes individuals that have certain disabilities, such as: color blindness, blindness, deafness and so on. Therefore, in time, the website should implement various tools which help with site navigation, such as screen readers and keyboard navigation for instance. Depending on the multitude of disabilities that people might be afflicted with, it is impossible to develop one-size fits all tools that suit every disabled individual from the start. Therefore, this is something that should be implemented over time based on the needs of those people that attend the school, and inevitably find the website lacking in some department.

15. **non-cryptographic threats and countermeasures**. Non-cryptanalytic attacks do not target the cryptographic algorithm or protocol itself. Instead, they more often than not attack availability or employ deceit in order to steal user credentials. Availability can be targeted with denial of service attacks. If firewall rules and architecture are set up properly, denial of service should pose no significant risk. A common non-cryptographic threat is social engineering or in plain english, fraud and deceit aimed at tricking a user to reveal their login credentials to malicious actors. This can be countered with two-factor authentication, which renders the credentials obsolete if one also doesn’t possess the second set of authentication credentials.

16. **cryptographic threats and countermeasures.** Cryptographic threats target either the confidentiality or the integrity of a protocol. Confidentiality can be broken by brute-force guessing keys. This is effectively stopped by the chosen algorithmic key lenghts. Integrity can be targeted with replay- and modification attacks for instance, which are countered by signed hashes and MACs (message authentication codes). These measures make sure that integrity is preserved, as was previously explained.