Milena Gonzalez

December 2018

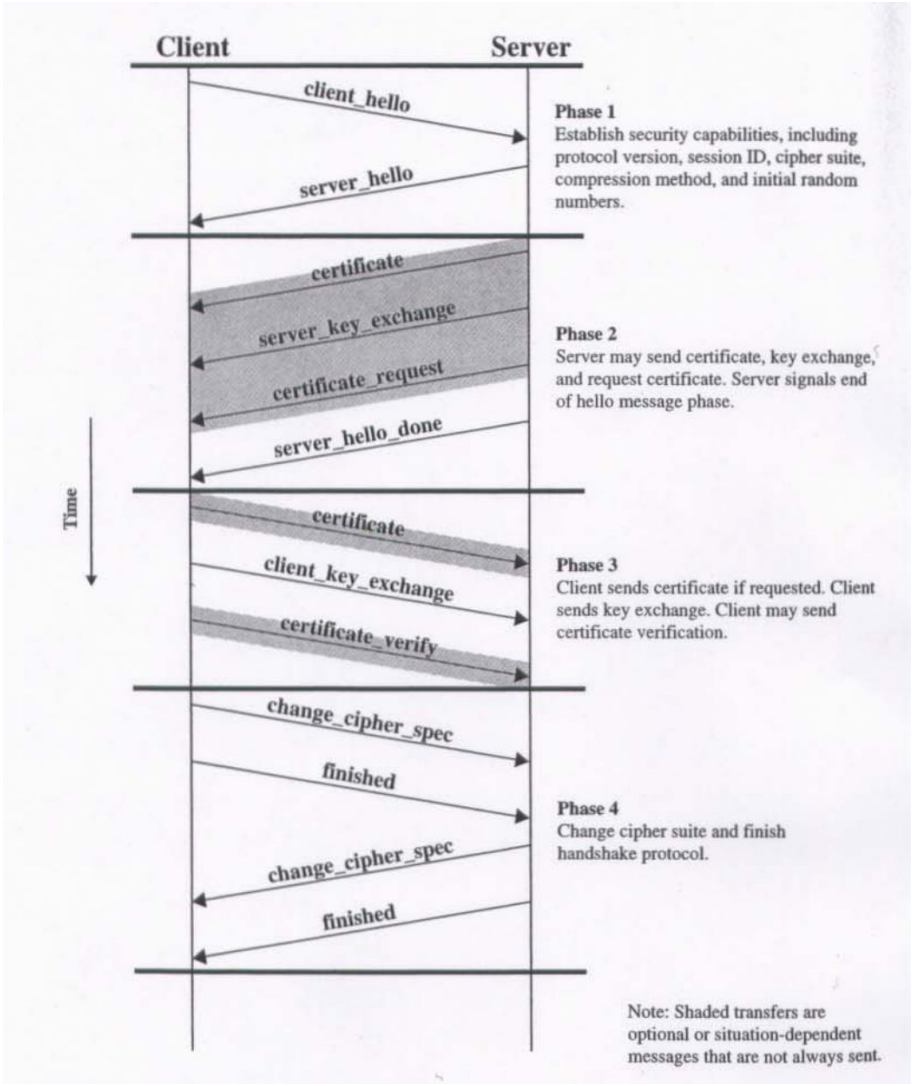
Cryptography and Network I

Final Project Write Up

Handshake Protocol

Our implementation of SSL Protocol focuses on fulfilling the 2 required services of SSL: Confidentiality and Message Integrity. Confidentiality is fulfilled by the shared key that is established during the Handshake Protocol. Message Integrity is established by the Message Authentication Code which uses the shared key as well.

The purpose of the Handshake Protocol is to allow the server and client to authenticate each other and to negotiate an encryption. This is used heavily everyday when a user needs to communicate with a modem, printer, or server. This protocol consists of a series of messages exchanged by client and server. The diagram to the right demonstrates the 4 phases of the Handshake protocol. The first service of Handshake Protocol is to establish security capabilities. This is when we initiate logical connection and establish security capabilities associated with it. The second task is the server authentication and the key exchange. The server key exchange message takes place at this time. A Certificate Message is



required for any agreed-on key exchange method. The third phase is the Client Authentication and Key Exchange. Now the client must initiate the client key exchange and verifies the certificate that the server sent contained the correct parameters. Now, the setup is secure! Now the client can send the change cipher spec message and copy the pending cipher spec to the current one.

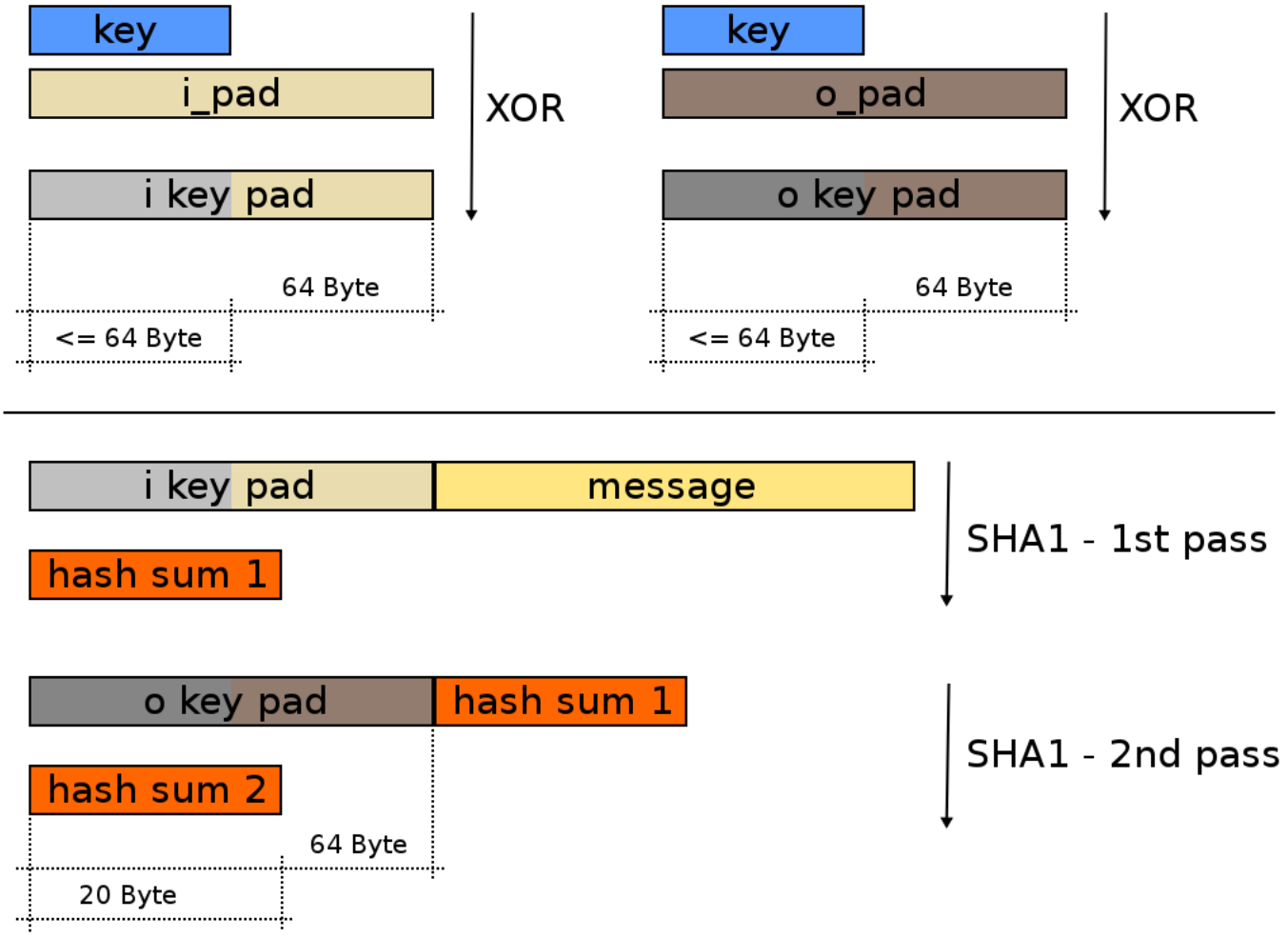
HMAC-SHA1

Regular data integrity checks are crucial to ensuring a secure connection. HMAC is a perfect example of such. HMAC stands for Keyed-Hashing for Message Authentication. MAC is established by a cryptographic hash function over the data that is to be authenticated and the shared key established previously. The HMAC enables communicating parties to verify the integrity and authenticity of the messages. Examples of secure file transfer protocols who achieve message authentication through HMAC include FTPS, SFTP, and HTTPS. A simpler way of putting it is that HMAC is just a container for hash functions, in this case SHA1, with a key to generate an authenticated digest.

We used HMAC-SHA1 is because of its efficiency. Hash functions can take a message of arbitrary length and transform it into a fixed length digest. So, even for large messages, corresponding digests can remain short, which maximizes bandwidth.

Through thorough research, I noticed that there was still a huge concern with HMAC-SHA1's independency on resistance to collisions. I found a suitable answer discussion the nature of targeting collisions on SHA1. To implement a collision attack on SHA1, one would be required to know the state of the SHA1 chaining variable is. Since the attacker doesn't know the shared key, they cannot predict this information.

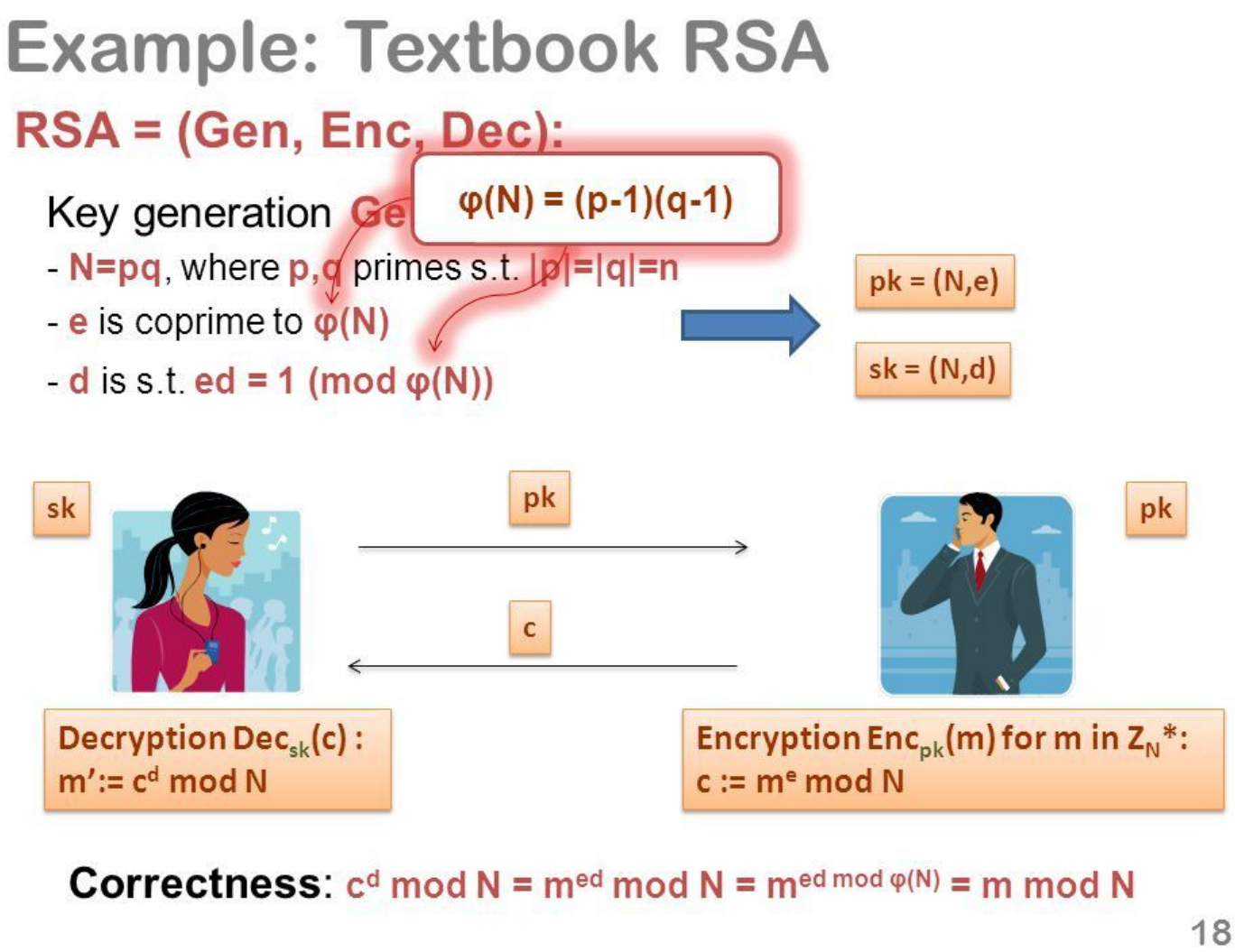
Below is a diagram demonstrating the flow of HMAC-SHA1.



RSA

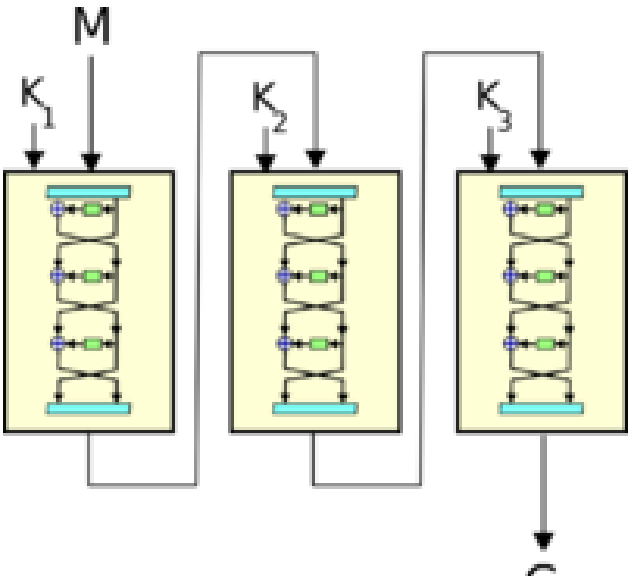
We implemented RSA for public key encryption in the KeyEcryption file, as well. The private key is used for authentication and a symmetric key exchange during establishment. RSA is part of the public key infrastructure.

For the most part, there does not appear to be any difference in the hardness of the RSA problem for different choices of the exponent e, as long as it isn't too small. But, we had to be careful to keep the private key relatively small. Every doubling of the RSA private key slows down the SSL handshake protocol about 6 or 7 times. The only thing is, we started trying to customize our RSA to build upon our SSL implementation, but I think we were a bit too ambitious. All of our components work fine on their own, we just had trouble incorporating them with our network services.



3DES

Triple DES was established after it was discovered that DES with a 56-bit key fails against brute force attacks. Triple DES was a simple solution to expanding the search space of the key without inventing a whole new algorithm. Triple DES is favored over double DES because it prevents against meet in the middle attacks while the latter does not. I simply expanded my Toy DES by increasing the sbox sizes, increased the rounds, and changing my code to handle larger messages. I also added the option to use plain DES, as well. I was surprised at



how slow the algorithm was. After doing further research, I realized this was not just my implementation. Triple DES is primarily used in hardware implementations (because it was designed for hardware) and is slowly being replaced in software by the Advanced Encryption Standard(AES). The reason 3DES is still believed to be secure at all is because it requires 2^112 operations which is not achievable with current technology. AES is considered the successor of DES in regards to the standard symmetric encryption algorithm. It is efficient in both software and hardware implementations. I was also surprised to learn that AES was developed during an open cryptography competition over several years.