
Homework #1
OREGON STATE UNIVERSITY
ECE 478 NETWORK SECURITY
SPRING 2016

Student:
Sam Quinn
Quinnsa@Oregonstate.edu

Professor:
Dr. Attila A Yavuz
Attila.Yavuz@oregonstate.edu

April 20, 2016



Contents

1	Question about basic concepts:	2
2	Data Encryption Standard (DES). DES has weak keys.	2
3	Block Cipher Design Principles and AES	3
4	Symmetric Encryption Modes	4
5	Ciphertext manipulability	5
6	Encryption and Compression	5
7	Symmetric key Authentication	6
8	Cryptographic hash functions	7
9	Length extension	8
10	Properties of cryptographic hash function	8

1 Question about basic concepts:

What are the main security/performance trade-offs for symmetric and asymmetric cryptography to be deployed in practice? Could you give real-life scenarios where symmetric and asymmetric crypto could be more suitable?

Symmetric and asymmetric cryptography are the two basic forms of encryption with each functioning differently than one another. Symmetric cryptography is faster but needs the secret key share between all parties before hand. Since there is one secret key that is shared between all parties before hand it all encryption as well as decryption is done with the same key, meaning that all parties use the same key. Symmetric is better when dealing with only trusted parties since you cannot establish authentication with only one secret key. Symmetric encryption is ideal for encrypting personal files where the secret key would not need to be shared.

Asymmetric cryptography dose not need the secret key to be shared before hand. Asymmetric encryption needs two keys, a private key and a public key. The public key may be shared with the presence of an eavesdropper with no security penalties. Asymmetric encryption keys do not need to be pre-shared with parties, and since every party has their own key pair it will also provide authentication. Asymmetric cryptography is ideal for applications where both parties have never spoken before, for example SSH.

OTPs are highly secure, but why we do not see them much in practice?

One time pad is one of the highest security since each bit of the plaintext is randomized, the random pattern used to in the OTP can only be used once or information can be extracted form the cipher text. The main problem with OTP is that the size of the key must be the size of the data, and a new key must be generated every message.

Does Kerckhoffs's principle contradict with the "secret algorithm" practice in military systems? Given sufficient financial capability, how could you incorporate Kerckhoffs's principle into such high-end systems?

The security systems used in the military are often classified until they come up with a new method in which the old systems are released to the public. Withholding the current system from the public introduces the property of security through obscurity since the adversary does not know the algorithms used. One way to introduce Kerckhoffs's principle in to military systems would be to hire a team of skilled cryptographers to study the security systems used thus giving them complete knowledge of the system minus the key. If the team are able to break the system then the only thing that is keeping the military security system secure is the obscurity which is very weak. However if the team is unable to break the system knowing everything but the key then the system is secure plus the added bonus of being obscure.

2 Data Encryption Standard (DES). DES has weak keys.

What is the difference between a weak key, a semi-weak key and a possible weak key?

The origin of weak, semi-weak, and possibly weak keys in DES start with how DES extracts each rounds encryption keys from the master 56bit key. The 56bit key is divided into 16 sub keys one for each cycle of the DES encryption. A weak key would be $\{0\}^{56}$ or $\{1\}^{56}$ where all of the bits are the same thus making the rotating of the keys for each cycle useless. A semi-weak key would be a key that would reverse the cycle that took place in the previous cycle. A key that repeats with a period of 2 meaning that the rotation of keys produces exactly two keys that will be each be used 8 times. Possibly weak keys are keys that repeat with a period of 4, outputting 4 unique keys that will all be used 4 times each. (<http://search.proquest.com/openview/6cc5dfe1f7c352582f2b62aa741b47dd/1?pq-origsite=gscholar>)

What is double DES? What kind of attack on double DES makes it useless?

Double DES does two rounds of DES encryption with two different keys, thus making the key 112 bits long. With a meet in the middle attack the added benefit of encrypting twice does not increase the security much. Meet in the middle attack. The effective security gain from the single DES 2^{56} is increased by one exponential to 2^{57} in double DES. Double DES is useless because an adversary could brute force the first round (single DES) and then just pass the output to the second round without the need for any brute forcing. (<http://stephanemoore.com/pdf/meetinthe middle.pdf>)

What is triple DES? How many keys are use in triple DES process?

Triple DES uses three rounds of DES making the maximum security 168 bits. Triple DES can either use 2 or 3 keys, however the effective security due to the meet in the middle attack explained above is 2^{112} for both. This is because if 2 of the 3 rounds are extract the adversary will be able to perform the meet in the middle attack on the first two and a normal brute force on the remaining round.

$$2^{56} + 2^{112} \approx 2^{112}$$

3 Block Cipher Design Principles and AES

What is the basic design technique, which is frequently used to construct modern symmetric ciphers?

Symmetric cryptography will use techniques to obscure the data with confusion and diffusion. Confusion is mainly accomplished by substitutions, replacing original data with new data. Diffusion is mainly accomplished by permutations, where one bit changes will permute on to all the ciphertext data to further scramble the original input. Modern symmetric ciphers will alternate substitutions and permutations.

What are the main security properties achieved via this designed technique?

Confusion in a symmetric cipher will help prevent attacks linear attacks. Because the data

is passed through a non-linear table often created by the secret key the data is translated in a non-linear way.

Diffusion will help prevent against pattern analysis attacks. Since every language has traceable patterns in grammar, diffusion makes these patterns much harder to spot. If one bit is changed the entire ciphertext will be changed not just the corresponding bit within the ciphertext.

Given the example of AES, which functional steps enable achieving these properties? Please provide specific names of these operations and briefly explain how they are applied in AES (all answers are brief for this question)?

- Confusion: S-Box - Rijndael S-Box is a lookup matrix generated by determining the multiplicative inverse of each byte from the original input.
- Diffusion: MixColumn - AES uses a MixColumn function on the Rijndael matrix that permutes any change with data through the use of XORs.

What are the benefits of the use of finite field arithmetic in AES?

Finite field arithmetic in AES is a redefinition of how to perform algebraic operations including multiplication, addition, subtraction, and division. A finite field has a finite number of integers that help prevent adversary against linear attacks. The prevention against linear attacks is achieved based on the fact that the linear arithmetic is abstracted to a point where there needs to be more computation before hand to unscramble the equations.

4 Symmetric Encryption Modes

We have discussed various Symmetric Encryption Modes. Each of these modes satisfies certain properties, which can be an advantage or disadvantage for a given application. Construct a table providing a summary information about each mode and its corresponding properties. For example, each row of the table will be properties (e.g., parallel operation, ciphertext manipulation, pre-computation, etc., please see course notes for more properties) and columns are Modes (e.g., CBC, CTR). Each cell will take a value such as Yes, No, partially, high, low etc. according to given encryption mode and property.

	ECB	CBC	OFB	CFB	CTR
Parallel	Yes	Decryption only	No	Decryption only	Yes
Leakage	Yes	No	No	No	No
Ciphertext Manipulation	Yes	No	No	No	No
Error Propagation	No	Yes	No	Yes	No
Pre-computation	No	No	Yes	No	Yes

5 Ciphertext manipulability

Ciphertext manipulability is generally considered as an undesirable property for Encryption Modes. However, for modes that operate in stream cipher fashion (discussed in class), by design, it is possible to flip bits in plaintext by flipping bits of ciphertext. Why is this possible?

These manipulations are possible based on the fundamental mathematics behind manipulable encryption algorithms. For RSA it is the fact that $C_1 * C_2 = X_1^e X_2^e \bmod(M)$, for ElGamal it is because of the cyclic group that the ciphertext will fall into.

Is there a way to turn this (potentially) undesirable property into an advantage, describe how if there is one?

Yes, when an encryption scheme has this property of allowing the ciphertext to effect the plaintext in a positive way it is called homomorphic. Elgamal is multiplicatively homomorphic, meaning you can perform multiplication operations on the ciphertext themselves that when decrypted will result in the same multiplication on the plaintext. So if we were given an unknown ciphertext $M \in G$ we can construct a different ciphertext that also decrypts to the same M . To get only the original M out in plaintext we can create a ciphertext c' with a random message M' . We can multiply the ciphertexts c and c' together to get $c * c'$. We can send $c * c'$ to a decryption oracle that will output the message $M * M'$, which all we would have to do is pull out the M' from the resulting plaintext. After we extract the message M' the original message M will be left over.

For good this property could be used to scale encrypted images stored in the cloud or perform mathematical operations over encrypted data without ever knowing the data itself.

Which extra cryptographic function (a group of functions discussed in the class) can be applied to the ciphertext so that the aforementioned advantage can be obtained without compromising the security?

Hints: (i) Consider noisy communication channels as application domain to exploit this feature. (ii) The extra cryptographic function will require annexing a small-constant tag to the ciphertext.

One way to ensure that the ciphertext has not been tampered with before passing it to the decryption oracle is to use an encryption scheme with authentication or encrypt the data then Authenticate E.g. a MAC. This would create a cryptographic checksum that would either be prepended or appended to the encrypted data itself so that the recipient could verify that the data had not been tampered with.

6 Encryption and Compression

Encryption (E) and Compression (C) are generally used together to achieve confidentiality and efficiency simultaneously. Given a message M , with which order function

E and C must be applied? What are the reasons behind of this particular order?

Compression must happen before encrypting. This is because compression algorithms usually will take advantage of patterns to minimize the data size, encryption is meant to make the output look random by eliminating patterns. With the encrypted data garbled most compression algorithms will not have anything to compress. However, if the data is compressed as plaintext the compression algorithms will function at the data's full compression availability which then can be encrypted after.

7 Symmetric key Authentication

In symmetric key cryptography, it is desirable to achieve both confidentiality and authentication (also provides integrity) of the data. These properties can be achieved via Encryption (E) and Authentication Functions (A), respectively. What is the correct order of these operations? Let's assume the specific notation and order of these operations are as follows:

- **Authenticate-then-Encrypt (AtE)** AtE will provide integrity on the plaintext but the cannot prevent against CPA until the recipient has decrypted the message to retrieve the MAC. In some encryption schemes both the ciphertext, MAC, and plaintext could be manipulated and reencapsulated since the MAC is located within the encryption, provided that the encryption scheme is malleable.
- **Encrypt-then-Authenticate (EtA)** EtA is the most useful order of operations since it will provide integrity of the ciphertext, which in turn also provides plaintext integrity. EtA will also prevent any CPA since any alterations to the ciphertext will be flagged as invalid.
- **Encryption and Authentication (E&A) or the opposite way as (A&E)** E&A will provide essentially the same benefits of EtA but since the plaintext is authenticated before encryption if the same message is being transmitted the MAC will be the same for every message duplicate message. As for A&E that is the same thing as E&A as those functions are happening simultaneously the order does not matter.

Discuss the security implications of these choices, which one is recommended and why?

The recommended order of operations would be encrypt then authenticate since will provide integrity of both the ciphertext as well as the plaintext. Since the MAC is taken after the encryption happens it will always produce a uniformly random MAC that would not reveal any information about the encrypted message.

Mention important crypto papers (at least one, cite it), in which the security of an important real-life protocol (Hint: the protocol that securely connects your VPN for each e-commerce transaction!) analyzed based on the above orders. Explain why this order matters a lot in practice? You may provide some discussions from these papers (please be brief, just hit on important points).

In the cryptography paper “The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)” they analyse common network communication method such as SSL, IPsec, and SSH with regards to how the order of operations effects the security.

SSL: $a = \text{Auth}(x), C = \text{Enc}(x, a), \text{transmit } C$

IPsec: $C = \text{Enc}(x), a = \text{Auth}(C), \text{transmit}(C, a)$

SSH: $C = \text{Enc}(x), a = \text{Auth}(x), \text{transmit}(C, a)$

EtA is considered secure so from this we can conclude that IPsec is considered secure. However, for SSL the security is questionable even when $\text{Enc}()$ is IND-CPA and $\text{Auth}()$ provides message unforgeability since the packet being transmitted is still susceptible to chosen ciphertext attacks. SSH uses E&A which is also not proven secure since the MAC while providing unforgeability can still leak information about the data being transferred [1].

8 Cryptographic hash functions

Given a modern cryptographic hash function (e.g., SHA) with m-bit length output, what is the maximum security it can provide in terms of m (lets call it x-bit security)? A generalized proof for any given m-bit hash function is available to show why it can achieve at best x-bit security. Please describe this generalized proof (related birthday attack concept) that simply connects m-bit to x-bit.

There must be collisions when there is data larger than the hash output. It relates to the pigeon hole theorem and birthday bounds where as the size of the data continues to increase the amount of potential collision rise. If a hash function returns a hash output size of $m - \text{bits}$ then to guarantee you will find a message you will need to hash 2^m random inputs. If you use the concept of the birthday paradox with at least 0.5 probability you will get a collision where two different messages have the same hash in $2^{m/2}$ random inputs.

To calculate the probability of a collision within a hash function the following formula is given.

$$\begin{aligned} & \Pr[h(m) = h(m')] \\ &= 1 - \Pr[h(x_1) = h_2] * \Pr[h(x_3) \notin h(x_1), h(x_2)] \cdots * \Pr[h(x_m) \notin h(x_1), \dots, h(x_{m-1})] \\ &= 1 - (1 - \frac{1}{2^n})(1 - \frac{1}{2^n}) \dots (1 - \frac{m-1}{2^n}) \\ &\approx 1 - e^{-\frac{1}{2^n} e^{\frac{-2}{2^n}} \dots e^{\frac{-(m-1)}{2^n}}} \end{aligned}$$

In the line above the value of e raised to the negative it is very close to zero we can't be sure what the absolute value will be so an approximation is needed to calculate.

$$\approx 1 - e^{\frac{-m^2}{2^{n+1}}}$$

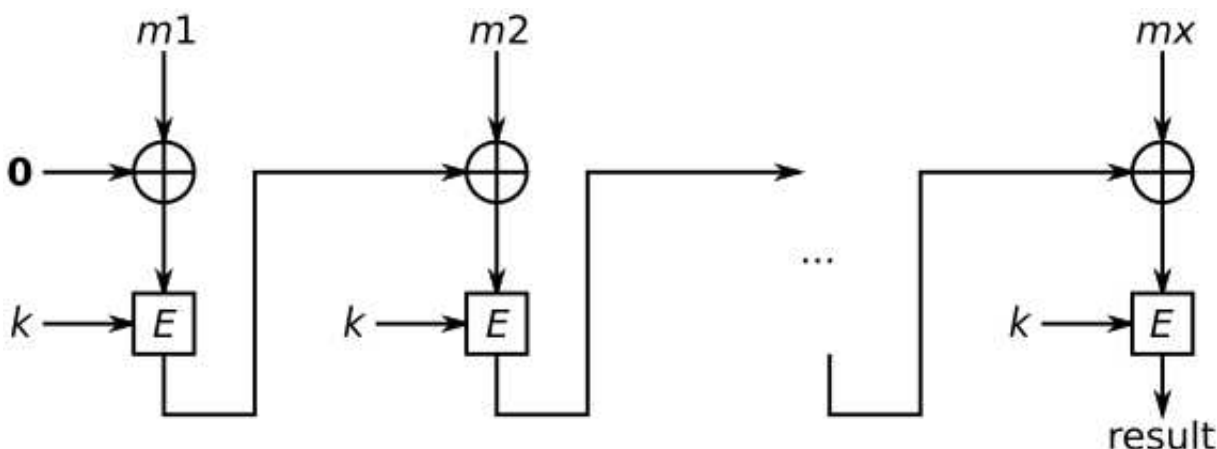
If we want to find a collision with at least 95% probability we can plug 95 in to our equation.

$$\begin{aligned} 0.95 &= 1 - e^{\frac{-m^2}{2^{n+1}}} \\ m^2 &= (-\ln 0.05) 2^{n+1} \\ m &\approx 2.45 * 2^{\frac{n}{2}} = O(2^{\frac{n}{2}}) \end{aligned}$$

9 Length extension

What is the hash length extension attack? Please describe by giving some specific real-life examples.

A length extension attack is an attack on HMACs with messages of varying length where an adversary may append new data to the authenticated message with the same signature.



CBC-MAC is susceptible to a length extension attack when the messages are not restricted to a single length. CBC-MAC will take the input message and compute the CBC encryption of the message. With the idea of a MAC to shorten the input to a fixed size, CBC-MAC will only export the last block of the message into its signature. As described above the output from e will be \otimes by m_x until the last block. An adversary is able to attack this MAC if they take the *result* from the original MAC m and $\otimes m'_1$ with the *result*. This would make the *result* cancel out the *result* \otimes that was in m'_1 . After we break the *result* with m'_1 we are able to append anything with in m' to the end of m with the same signature.

$$\begin{aligned} CBC_MAC(k, m_1, \dots, m_l) &\rightarrow (m, t) \\ CBC_MAC(k, m'_1 \otimes t, \dots, m'_l) &\rightarrow (m \| m', t') \end{aligned}$$

(<https://cseweb.ucsd.edu/~mihir/papers/cbc.pdf>)

10 Properties of cryptographic hash function

What are the essential properties that a cryptographic hash function must satisfy?

- **Performance:** Must be easy to compute $H(m)$

- **One-way property:** Given $H(m)$ should not be able to find m
- **Weak/Strong Collision Resistance:** $H(m) \neq H(m')$ and $H(m_1) \neq H(m_2)$ if $m_1 \neq m_2$

What is a Random Oracle and how does it play a role in the security proofs in general (what is its relation with cryptographic hash functions?)

In security proofs a random oracle is used as a baseline of what the proposed cryptographic function should be indistinguishable against. The random oracle will always produce uniformly random output that has nothing to do with the data passed into the “blackbox”. When proving an encryption algorithm or a cryptographic function is secure there should be no way for an adversary to determine if the output came from either the random oracle or the encryption scheme. If an adversary is able to exploit the cryptographic function to a point where they will get deterministic output, then the cryptographic function would not be considered secure.

Hash functions security can be proven with the use of a random oracle. If an adversary can modify the plaintext input to produce an output that looks anything but uniformly random then the hash function should not be considered secure. The main limitation with a random oracle in regards to cryptographic hash function is that the output is deterministic, meaning that given the same input it should produce the same output. The random oracle can mimic this feature by creating a lookup table of input and their respective outputs. If a new input comes in the random oracle will return a uniformly random output, but store the output with an association to the input. So if the same input is ever passed in the random oracle will return the same uniformly random string as the first time.

References

- [1] H. Krawczyk, “The order of encryption and authentication for protecting communications (or: How secure is ssl?),” in *Advances in Cryptology CRYPTO 2001*, pp. 310–331, Springer, 2001.