Introduction to Cryptography

Willie Agnew

GT Big O Theoretical Computer Science Club, 2015

Outline

Outline

What is Cryptography?

- Cryptography is the secure manipulation of data in the presence of adversaries. While this manipulation has traditionally been transmitting data, recently cryptosystems that can do other things, like compute on data, have been created.

Cryptography In the World

- During WWII, Alan Turing and other cryptographers broke German codes shortening WWII by several years and saving millions of lives.
- The NSA is known/suspected to have broken several cryptographic primitives.

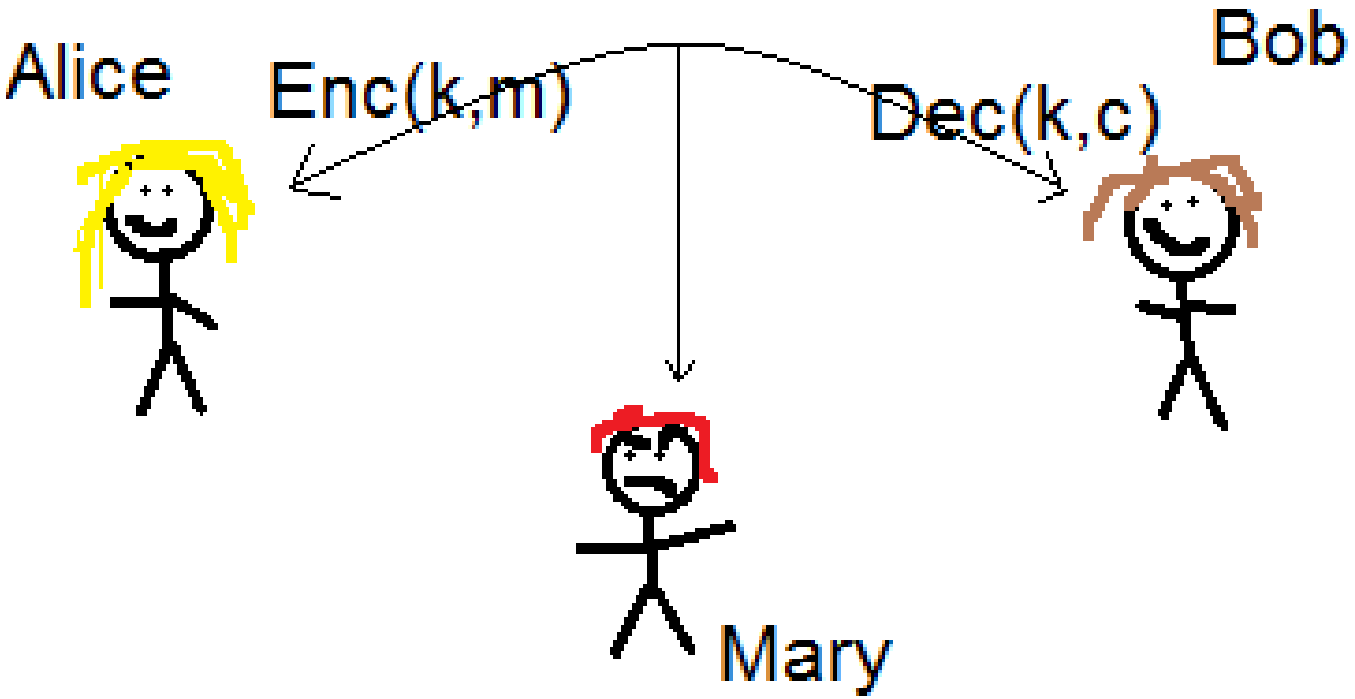
Cryptography In Everyday Life

- Cryptography is used to secure internet communications: Facebook, email, banking, anything that involves sending sensitive information.
- Without cryptography anyone could see your Snapchats or even impersonate you on Facebook.

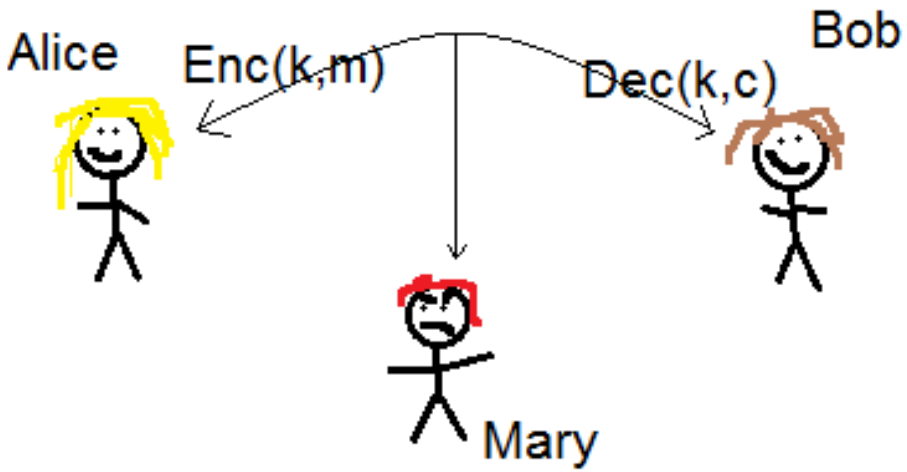
Outline

Encryption Scenario

- Alice and Bob want to communicate securely over a distance. Mary is a creeper and will do her best to eavesdrop on them, so they first get together and create a secret, s. Then they move apart and start sending messages, m, using a key $Gen(s) = k$ derived from the secret and some encryption and decryption algorithms, $Enc(k, m) = c$ and $Dec(k, c) = m$. Mary can only read sent messages. Alice and Bob don't want Mary to learn anything about the contents of the messages.

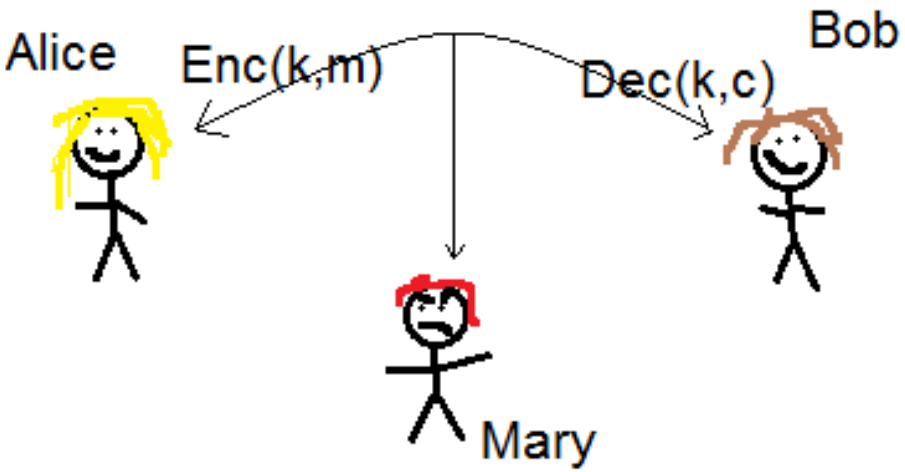


Encryption Properties: IND



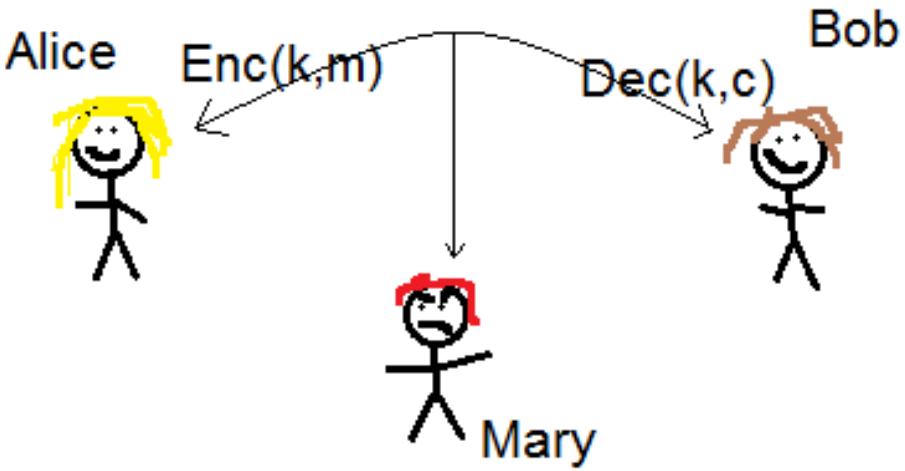
- What properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?

Encryption Properties: IND



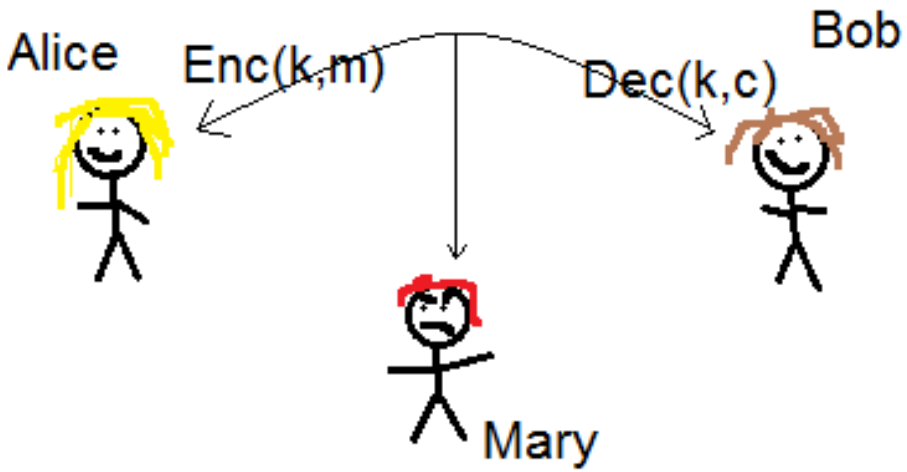
- What properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?
- $Dec(k,\ Enc(k,\ m)) = m$

Encryption Properties: IND



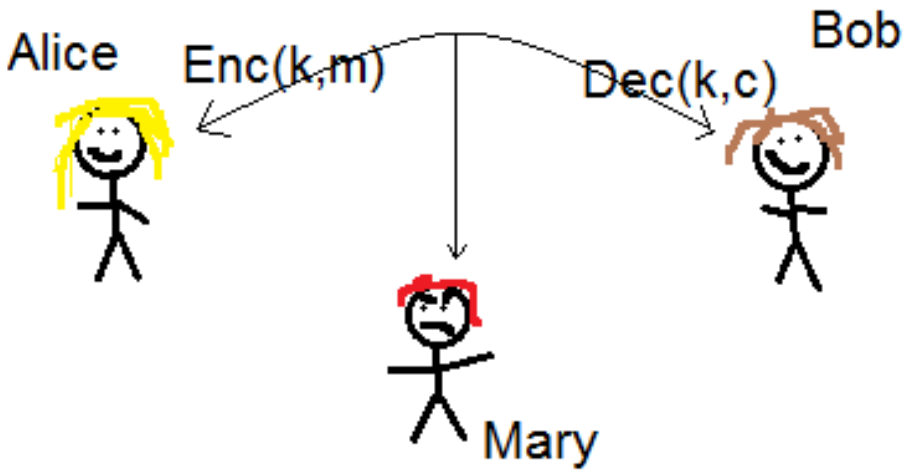
- What properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?
- $Dec(k,\ Enc(k,\ m)) = m$
- $$\Pr[A(m, Enc(k,\ m), Enc(k, m')) =\ m] \leq 1/2 + negl(|k|)$$

Encryption Properties: IND



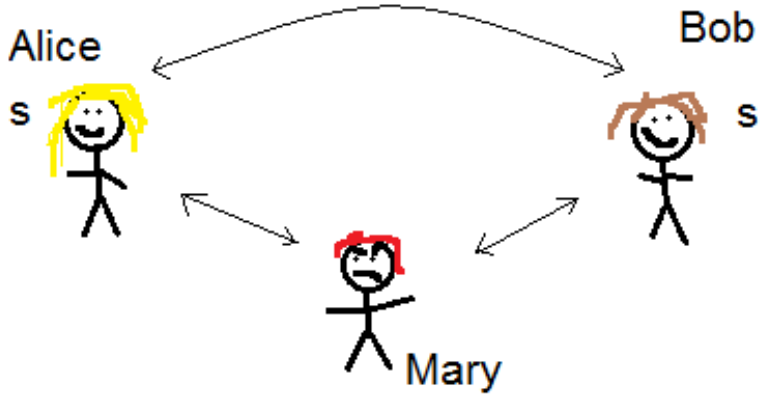
- What properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?
- $Dec(k,\ Enc(k,\ m)) = m$
- $$\Pr[A(m, Enc(k,\ m), Enc(k, m')) =\ m] \leq 1/2 + negl(|k|)$$
- This means that the chance the ppt adversary A (Mary) can guess which of two encryptions under k, $Enc(k,\ m)$ and $Enc(k,\ m')$, $m \neq m'$ is the encryption of m is less than $1/2$ plus a term negligibly (exponentially) small in the length of the key, $|k|$. That is, the adversary cannot efficiently distinguish between encryptions of different messages.

Encryption Properties: IND



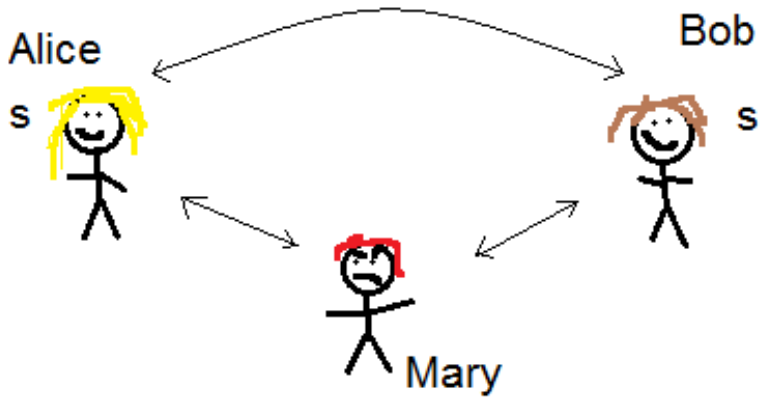
- What properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?
- $Dec(k,\ Enc(k,\ m)) = m$
- $$\Pr[A(m, Enc(k,\ m), Enc(k, m')) =\ m] \leq 1/2 + negl(|k|)$$
- This means that the chance the ppt adversary A (Mary) can guess which of two encryptions under k, $Enc(k,\ m)$ and $Enc(k,\ m')$, $m \neq m'$ is the encryption of m is less than $1/2$ plus a term negligibly (exponentially) small in the length of the key, $|k|$. That is, the adversary cannot efficiently distinguish between encryptions of different messages.
- All algorithms have a source of randomness as an implicit input, and all adversaries have a source of randomness and the key size as implicit inputs.

Encryption Properties: IND-CPA



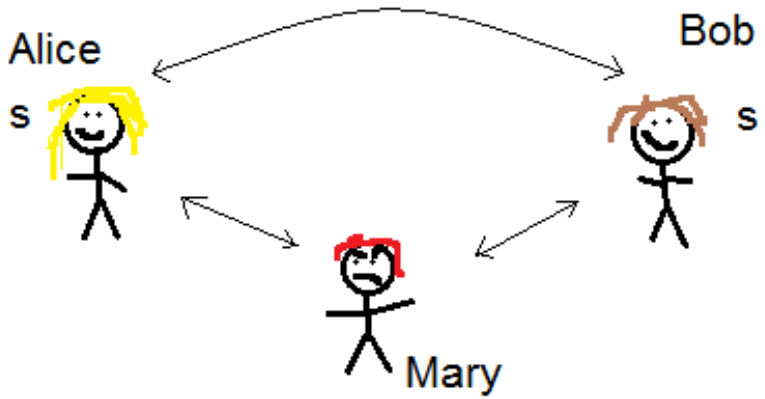
- Mary can now trick Alice and Bob into encrypting messages for her.
- IRL scenario: Mary emails a message to Alice. Alice stores this email on her encrypted hard drive. Mary steals Alice's hard drive and now has the encryption of her message.
- What new properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?

Encryption Properties: IND-CPA



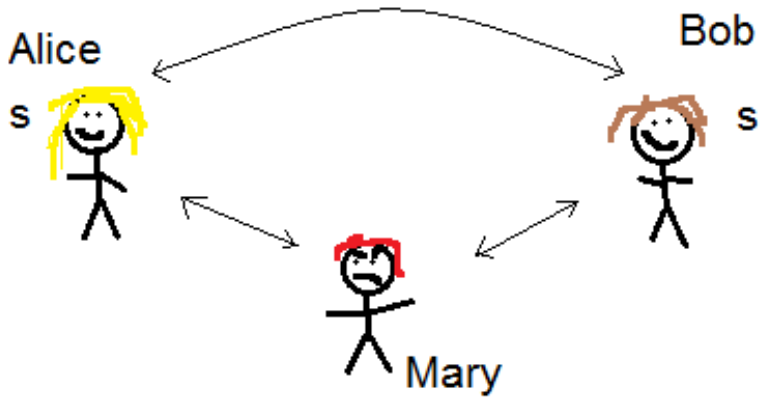
- Mary can now trick Alice and Bob into encrypting messages for her.
- IRL scenario: Mary emails a message to Alice. Alice stores this email on her encrypted hard drive. Mary steals Alice's hard drive and now has the encryption of her message.
- What new properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?
- $$\Pr[A(C^b(k,\ m_0,\ m_1)) = b \leq 1/2 + negl(|k|)$$

Encryption Properties: IND-CPA



- Mary can now trick Alice and Bob into encrypting messages for her.
- IRL scenario: Mary emails a message to Alice. Alice stores this email on her encrypted hard drive. Mary steals Alice's hard drive and now has the encryption of her message.
- What new properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?
- $$\Pr[A(C^b(k,\ m_0,\ m_1)) = b \leq 1/2 + negl(|k|)$$
- $C^b(k,\ m_0,\ m_1)$ is an encryption oracle that, when queried with messages m_0 and m_1 returns $Enc(k,\ m_b)$, $b \in \{0,\ 1\}$. A can query this oracle as many times as it likes (within its ppt runtime bound).

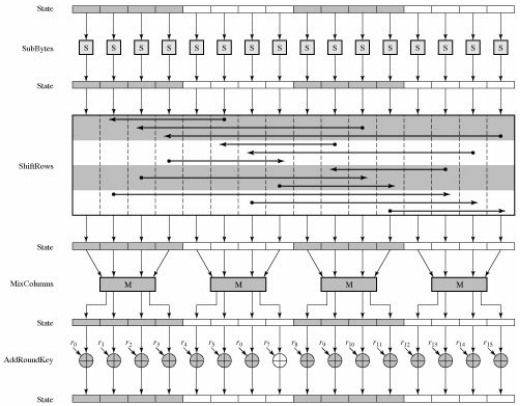
Encryption Properties: IND-CPA

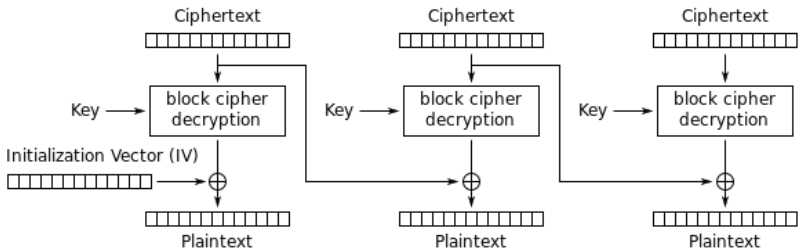


- Mary can now trick Alice and Bob into encrypting messages for her.
- IRL scenario: Mary emails a message to Alice. Alice stores this email on her encrypted hard drive. Mary steals Alice's hard drive and now has the encryption of her message.
- What new properties must our encryption algorithm, $\{Gen, Enc, Dec\}$, have?
- $$\Pr[A(C^b(k,\ m_0,\ m_1)) = b \leq 1/2 + negl(|k|)$$
- $C^b(k,\ m_0,\ m_1)$ is an encryption oracle that, when queried with messages m_0 and m_1 returns $Enc(k,\ m_b)$, $b \in \{0,\ 1\}$. A can query this oracle as many times as it likes (within its ppt runtime bound).
- How does this new definition capture Mary's ability to obtain the encryptions of any messages she wants?

Implementing an IND-CPA secure cipher in Java

- There are theoretically secure ciphers, but these are slow.
- In practice "secure" ciphers are used. AES is the most popular of these. To prevent one time pad attacks, the AES-CBC and AES-GCM constructions are used.
- We will implement AES-CBC with a 128-bit key. In theory, this should provide 128 bits of security; that is, it will take an attacker $\mathcal{O}(2^{128})$ operations to decrypt a message.
- In reality, AES is "broken" and there are attacks that take $\mathcal{O}(2^{126.1})$ operations to decrypt a message. To put this in perspective, it would take the Tianhe-2, the most powerful supercomputer in the world at 33,862.7 TFLOPS/s, over 85 billion years to run this attack even if each AES operation was only 1 FLOP (they are much more).





Cipher Block Chaining (CBC) mode decryption

Outline

Are We Secure Using Just a Cipher?

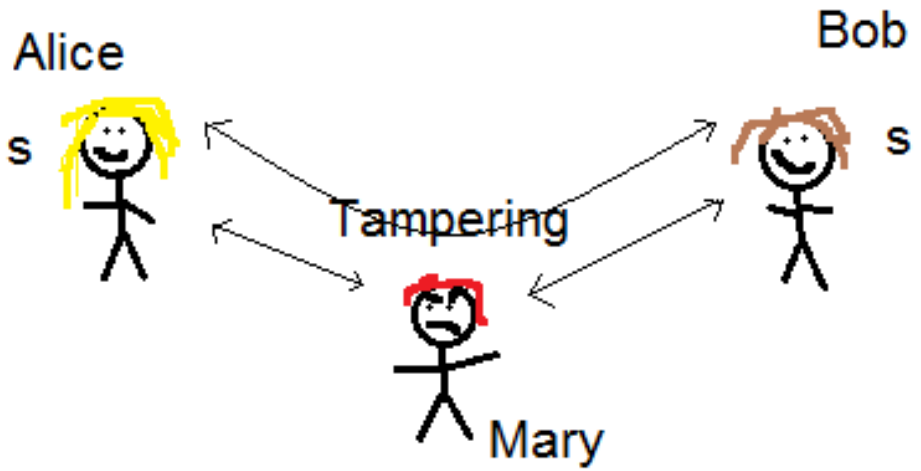
- Recall our model. Bob and Alice send messages back and forth. Mary attempts to gain some information about these messages by reading but not changing Alice's and Bob's communications. What is unrealistic about this model?

Are We Secure Using Just a Cipher?

- Recall our model. Bob and Alice send messages back and forth. Mary attempts to gain some information about these messages by reading but not changing Alice's and Bob's communications. What is unrealistic about this model?
- IRL, intercepting and modifying messages isn't hard, so we must assume Mary can do so.

Are We Secure Using Just a Cipher?

- Recall our model. Bob and Alice send messages back and forth. Mary attempts to gain some information about these messages by reading but not changing Alice's and Bob's communications. What is unrealistic about this model?
- IRL, intercepting and modifying messages isn't hard, so we must assume Mary can do so.



The Dangers

- If you don't ensure the authenticity and integrity of your messages *all of your friends will hate you*.

MACs

- How do we handle Mary tampering with messages?

MACs

- How do we handle Mary tampering with messages?
- We should try to detect when a message has been tampered with.

MACs

- How do we handle Mary tampering with messages?
- We should try to detect when a message has been tampered with.
- Consider two algorithms, Tag and Ver. $Tag(k', m) = t$, with $|m| >> |a|$, that is, *Tag* outputs some identification tag for m. $Ver(k', m, t) = 1$, that is, it outputs true if the tag is valid for the given message and key. What properties must *Tag* and *Ver* have?

MACs

- How do we handle Mary tampering with messages?
- We should try to detect when a message has been tampered with.
- Consider two algorithms, Tag and Ver. $Tag(k', m) = t$, with $|m| >> |a|$, that is, Tag outputs some identification tag for m. $Ver(k', m, t) = 1$, that is, it outputs true if the tag is valid for the given message and key. What properties must Tag and Ver have?
- Mary (A) must not be able to forge a tag for a new message, even when given access to a tagging oracle:

$$\Pr[A^{Tag_k}() \text{ outputs } (m, t) \text{ s.t. } Ver(m, t) = 1 \leq negl(|k'|)]$$

MACs

- How do we handle Mary tampering with messages?
- We should try to detect when a message has been tampered with.
- Consider two algorithms, Tag and Ver. $Tag(k', m) = t$, with $|m| >> |a|$, that is, *Tag* outputs some identification tag for m. $Ver(k', m, t) = 1$, that is, it outputs true if the tag is valid for the given message and key. What properties must *Tag* and *Ver* have?
- Mary (A) must not be able to forge a tag for a new message, even when given access to a tagging oracle:

$$\Pr[A^{Tag_k}() \textit{ outputs } (m, t) \textit{ s.t. } Ver(m, t) = 1 \leq negl(|k'|)]$$

- When we require that $(m, t) \neq (m', t')$ but not $t \neq t'$ and $m \neq m'$, our *MAC* has strong perfect unforgeability under choosen message attack (UF-CMA).

MACs

- How do we handle Mary tampering with messages?
- We should try to detect when a message has been tampered with.
- Consider two algorithms, Tag and Ver. $Tag(k', m) = t$, with $|m| >> |a|$, that is, *Tag* outputs some identification tag for m. $Ver(k', m, t) = 1$, that is, it outputs true if the tag is valid for the given message and key. What properties must *Tag* and *Ver* have?
- Mary (A) must not be able to forge a tag for a new message, even when given access to a tagging oracle:

$$\Pr[A^{Tag_k}() \text{ outputs } (m, t) \text{ s.t. } Ver(m, t) = 1 \leq negl(|k'|)]$$

- When we require that $(m, t) \neq (m', t')$ but not $t \neq t'$ and $m \neq m'$, our *MAC* has strong perfect unforgeability under choosen message attack (UF-CMA).
- How many bits of security does a $|t| = b$ bit tag give us?

MACs

- How do we handle Mary tampering with messages?
- We should try to detect when a message has been tampered with.
- Consider two algorithms, Tag and Ver. $Tag(k', m) = t$, with $|m| >> |a|$, that is, *Tag* outputs some identification tag for m. $Ver(k', m, t) = 1$, that is, it outputs true if the tag is valid for the given message and key. What properties must *Tag* and *Ver* have?
- Mary (A) must not be able to forge a tag for a new message, even when given access to a tagging oracle:

$$\Pr[A^{Tag_k}() \textit{ outputs } (m, t) \textit{ s.t. } Ver(m, t) = 1 \leq negl(|k'|)]$$

- When we require that $(m, t) \neq (m', t')$ but not $t \neq t'$ and $m \neq m'$, our *MAC* has strong perfect unforgeability under choosen message attack (UF-CMA).
- How many bits of security does a $|t| = b$ bit tag give us?
- We call the two algorithms $\{Tag, Ver\}$ a MAC (Message Authentication Code).

MACs

- How do we handle Mary tampering with messages?
- We should try to detect when a message has been tampered with.
- Consider two algorithms, Tag and Ver. $Tag(k', m) = t$, with $|m| >> |a|$, that is, *Tag* outputs some identification tag for m. $Ver(k', m, t) = 1$, that is, it outputs true if the tag is valid for the given message and key. What properties must *Tag* and *Ver* have?
- Mary (A) must not be able to forge a tag for a new message, even when given access to a tagging oracle:

$$\Pr[A^{Tag_k}() \text{ outputs } (m, t) \text{ s.t. } Ver(m, t) = 1 \leq negl(|k'|)]$$

- When we require that $(m, t) \neq (m', t')$ but not $t \neq t'$ and $m \neq m'$, our *MAC* has strong perfect unforgeability under choosen message attack (UF-CMA).
- How many bits of security does a $|t| = b$ bit tag give us?
- We call the two algorithms $\{Tag, Ver\}$ a MAC (Message Authentication Code).
- If we successfully combine a strong UF-CMA MAC and an IND-CPA secure cipher we obtain an authenticated encryption scheme.

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.
- (Universal) Hash functions that are used:

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.
- (Universal) Hash functions that are used:
  - MD4 (1990): Collisions can be found in $\leq$ two hashes.

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.
- (Universal) Hash functions that are used:
  - MD4 (1990): Collisions can be found in $\leq$ two hashes.
  - MD5 (1992): Collisions can be found in $\mathcal{O}(2^{18})$ time.

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.
- (Universal) Hash functions that are used:
  - MD4 (1990): Collisions can be found in $\leq$ two hashes.
  - MD5 (1992): Collisions can be found in $\mathcal{O}(2^{18})$ time.
  - SHA-1 (1995): Collisions can be found in $\mathcal{O}(2^{60.3}) - \mathcal{O}(2^{65.3})$ time.

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.
- (Universal) Hash functions that are used:
  - MD4 (1990): Collisions can be found in $\leq$ two hashes.
  - MD5 (1992): Collisions can be found in $\mathcal{O}(2^{18})$ time.
  - SHA-1 (1995): Collisions can be found in $\mathcal{O}(2^{60.3}) - \mathcal{O}(2^{65.3})$ time.
  - SHA-2 (2001): Still considered secure, but of similar construction to SHA-1.

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.
- (Universal) Hash functions that are used:
  - MD4 (1990): Collisions can be found in $\leq$ two hashes.
  - MD5 (1992): Collisions can be found in $\mathcal{O}(2^{18})$ time.
  - SHA-1 (1995): Collisions can be found in $\mathcal{O}(2^{60.3}) - \mathcal{O}(2^{65.3})$ time.
  - SHA-2 (2001): Still considered secure, but of similar construction to SHA-1.
  - SHA-3 (Standardized August 5, 2015)

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.
- (Universal) Hash functions that are used:
  - MD4 (1990): Collisions can be found in $\leq$ two hashes.
  - MD5 (1992): Collisions can be found in $\mathcal{O}(2^{18})$ time.
  - SHA-1 (1995): Collisions can be found in $\mathcal{O}(2^{60.3}) - \mathcal{O}(2^{65.3})$ time.
  - SHA-2 (2001): Still considered secure, but of similar construction to SHA-1.
  - SHA-3 (Standardized August 5, 2015)
- So are these hash functions our MACs?

Implementing Authenticated Encryption

- Similar to ciphers, there are theoretically secure MAC constructions, but they are slow.
- (Universal) Hash functions that are used:
  - MD4 (1990): Collisions can be found in $\leq$ two hashes.
  - MD5 (1992): Collisions can be found in $\mathcal{O}(2^{18})$ time.
  - SHA-1 (1995): Collisions can be found in $\mathcal{O}(2^{60.3}) - \mathcal{O}(2^{65.3})$ time.
  - SHA-2 (2001): Still considered secure, but of similar construction to SHA-1.
  - SHA-3 (Standardized August 5, 2015)
- So are these hash functions our MACs?
- No, but we can incorporate our key, k' into them using the HMAC construction: $MAC(k',\ m) = HMAC(k',\ m) = H((k' \oplus 0x5c5c...5c)||H((k' \oplus 0x3636...36)||m))$.

Implementing Authenticated Encryption: Combining Ciphers and MACs

- How do we combine our cipher with our MAC?

Implementing Authenticated Encryption: Combining Ciphers and MACs

- How do we combine our cipher with our MAC?
- MAC-and-Encrypt: $MAC(m)||Cipher(m)$?

Implementing Authenticated Encryption: Combining Ciphers and MACs

- How do we combine our cipher with our MAC?
- MAC-and-Encrypt: $MAC(m)||Cipher(m)$?
- Encrypt-then-MAC: $MAC(Cipher(m))||Cipher(m)$?

Implementing Authenticated Encryption: Combining Ciphers and MACs

- How do we combine our cipher with our MAC?
- MAC-and-Encrypt: $MAC(m)||Cipher(m)$?
- Encrypt-then-MAC: $MAC(Cipher(m))||Cipher(m)$?
- MAC-then-Encrypt: $Cipher(MAC(m)||m)$?

Implementing Authenticated Encryption: Combining Ciphers and MACs

- How do we combine our cipher with our MAC?
- MAC-and-Encrypt: $MAC(m)||Cipher(m)$?
- Encrypt-then-MAC: $MAC(Cipher(m))||Cipher(m)$?
- MAC-then-Encrypt: $Cipher(MAC(m)||m)$?
- MAC-and-Encrypt: Bad, nothing in our definition of a MAC says it must reveal nothing about the message it tagged.

Implementing Authenticated Encryption: Combining Ciphers and MACs

- How do we combine our cipher with our MAC?
- MAC-and-Encrypt: $MAC(m)||Cipher(m)$?
- Encrypt-then-MAC: $MAC(Cipher(m))||Cipher(m)$?
- MAC-then-Encrypt: $Cipher(MAC(m)||m)$?
- MAC-and-Encrypt: Bad, nothing in our definition of a MAC says it must reveal nothing about the message it tagged.
- Encrypt-then-MAC: Good, even guarantees ciphertext authenticity and integrity.

Implementing Authenticated Encryption: Combining Ciphers and MACs

- How do we combine our cipher with our MAC?
- MAC-and-Encrypt: $MAC(m)||Cipher(m)$?
- Encrypt-then-MAC: $MAC(Cipher(m))||Cipher(m)$?
- MAC-then-Encrypt: $Cipher(MAC(m)||m)$?
- MAC-and-Encrypt: Bad, nothing in our definition of a MAC says it must reveal nothing about the message it tagged.
- Encrypt-then-MAC: Good, even guarantees ciphertext authenticity and integrity.
- MAC-then-Encrypt: Bad, for subtle reasons. Ciphertext integrity is not assured, which gives the attacker extra leeway in what she can do.

Outline

Is our Model Reasonable?

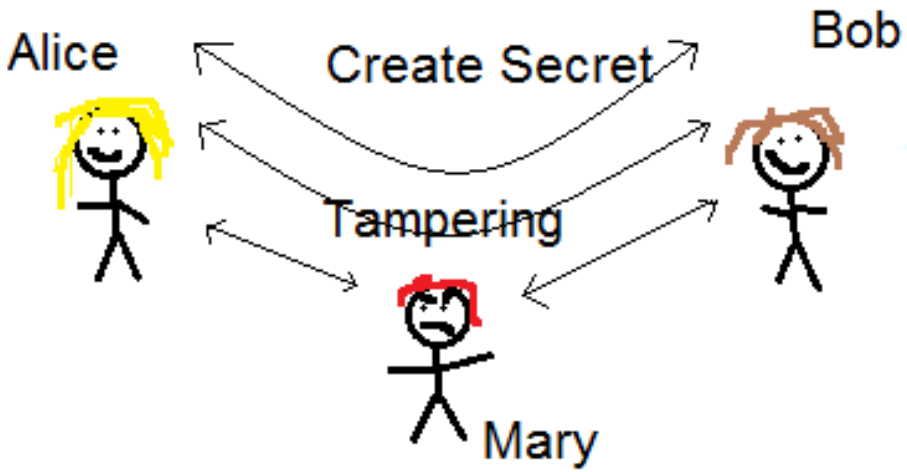
- Recall our model. Alice and Bob get together and share a secret. Then they use this secret to establish authenticated encryption. Mary can read and modify their communications and obtain the encryptions of any messages she wants. Is this a reasonable model?

Is our Model Reasonable?

- Recall our model. Alice and Bob get together and share a secret. Then they use this secret to establish authenticated encryption. Mary can read and modify their communications and obtain the encryptions of any messages she wants. Is this a reasonable model?
- No, IRL people aren't going to be able to get together and share a secret every time they want to communicate.

Is our Model Reasonable?

- Recall our model. Alice and Bob get together and share a secret. Then they use this secret to establish authenticated encryption. Mary can read and modify their communications and obtain the encryptions of any messages she wants. Is this a reasonable model?
- No, IRL people aren't going to be able to get together and share a secret every time they want to communicate.



Secret Sharing

- We need a way for Alice and Bob to establish a shared secret even with an eavesdropping, message-tampering Mary. That is, Mary "can't" determine the shared secret in ppt time.
- What does "can't" mean? Theoretical cryptography works as follows: problem X is "hard" to solve. You prove that if an adversary A can break the security of your cryptosystem, then A can be used to efficiently solve X. But since X "can't" be solved efficiently, A "can't" exist, so your cryptosystem is "secure."
- In practice, problem X is integer factorization. However, since quantum computers can efficiently factorize integers, cryptosystems based on other "hard" problems, including lattice problems like SIS, are being developed.

Diffie-Hellman

- So, what is a public key exchange protocol based on the hardness of integer factorization (Not RSA, it is too slow and requires large keys)?

Diffie-Hellman

- So, what is a public key exchange protocol based on the hardness of integer factorization (Not RSA, it is too slow and requires large keys)?
- Hint: Alice has a, g, and g^a. Bob has b, g, and g^b.

Diffie-Hellman

- So, what is a public key exchange protocol based on the hardness of integer factorization (Not RSA, it is too slow and requires large keys)?
- Hint: Alice has a, g, and g^a. Bob has b, g, and g^b.
- Diffie-Hellman key exchange: Alice sends g^a which Bob uses to compute $(g^a)^b = g^{ab}$, the shared secret. Then Bob sends g^b, which Alice uses to compute $(g^b)^a = g^{ab}$.

Diffie-Hellman

- So, what is a public key exchange protocol based on the hardness of integer factorization (Not RSA, it is too slow and requires large keys)?
- Hint: Alice has a, g, and g^a. Bob has b, g, and g^b.
- Diffie-Hellman key exchange: Alice sends g^a which Bob uses to compute $(g^a)^b = g^{ab}$, the shared secret. Then Bob sends g^b, which Alice uses to compute $(g^b)^a = g^{ab}$.
-

Party:	Alice	Mary	Bob
Start:	a, g, g^a	g	b, g, g^b
Alice Sends g^a:	a, g, g^a	g, g^a	b, g, g^b, g^a
Bob Sends g^b:	a, g, g^a, g^b	g, g^a, g^b	b, g, g^b, g^a
Alice and Bob Compute g^{ab}:	a, g, g^a, g^b, g^{ab}	g, g^a, g^b	b, g, g^b, g^a, g^{ab}

Diffie-Hellman

- So, what is a public key exchange protocol based on the hardness of integer factorization (Not RSA, it is too slow and requires large keys)?
- Hint: Alice has a, g, and g^a. Bob has b, g, and g^b.
- Diffie-Hellman key exchange: Alice sends g^a which Bob uses to compute $(g^a)^b = g^{ab}$, the shared secret. Then Bob sends g^b, which Alice uses to compute $(g^b)^a = g^{ab}$.
-

Party:	Alice	Mary	Bob
Start:	a, g, g^a	g	b, g, g^b
Alice Sends g^a:	a, g, g^a	g, g^a	b, g, g^b, g^a
Bob Sends g^b:	a, g, g^a, g^b	g, g^a, g^b	b, g, g^b, g^a
Alice and Bob Compute g^{ab}:	a, g, g^a, g^b, g^{ab}	g, g^a, g^b	b, g, g^b, g^a, g^{ab}

- If Mary can factor, she can retrieve a from g^a or b from g^b, allowing her to compute the shared secret, g^{ab}.

Diffie-Hellman

- So, what is a public key exchange protocol based on the hardness of integer factorization (Not RSA, it is too slow and requires large keys)?
- Hint: Alice has a, g, and g^a. Bob has b, g, and g^b.
- Diffie-Hellman key exchange: Alice sends g^a which Bob uses to compute $(g^a)^b = g^{ab}$, the shared secret. Then Bob sends g^b, which Alice uses to compute $(g^b)^a = g^{ab}$.
-

Party:	Alice	Mary	Bob
Start:	a, g, g^a	g	b, g, g^b
Alice Sends g^a:	a, g, g^a	g, g^a	b, g, g^b, g^a
Bob Sends g^b:	a, g, g^a, g^b	g, g^a, g^b	b, g, g^b, g^a
Alice and Bob Compute g^{ab}:	a, g, g^a, g^b, g^{ab}	g, g^a, g^b	b, g, g^b, g^a, g^{ab}

- If Mary can factor, she can retrieve a from g^a or b from g^b, allowing her to compute the shared secret, g^{ab}.
- Notice that Alice does not receive b and Bob does not receive a: these remain secret, allowing Alice and Bob to reuse g^a and g^b.

Implementing Diffie-Hellman

- In practice ECDH (Elliptic Curve Diffie-Hellman) is now used as a key exchange machanism. It is slightly faster and requires transmitting a bit less information than Diffie-Hellman.
- Use ECDH to create a shared secret between Alice and Bob. Then use a secure hash function (SHA-256) to derive *different* (any correlations between keys potentially reduces the size of the keyspaces that an attacker must search) key bits for the AES key, AES IV, and HMAC key.

Problems with Public-Key Exchange

- Our protocol has a serious flaw.

Problems with Public-Key Exchange

- Our protocol has a serious flaw.
- Mary can tamper with the Diffie-Hellman key exchange messages and establish her own shared secrets with Alice and Bob! This is known as a man-in-the-middle attack. How can we prevent this?

Problems with Public-Key Exchange

- Our protocol has a serious flaw.
- Mary can tamper with the Diffie-Hellman key exchange messages and establish her own shared secrets with Alice and Bob! This is known as a man-in-the-middle attack. How can we prevent this?
- Certificates, which give a static public key verified by a trusted third party (Certificate Authority, or CA, whose public key is known) are a widely used solution.

Problems with Public-Key Exchange

- Our protocol has a serious flaw.
- Mary can tamper with the Diffie-Hellman key exchange messages and establish her own shared secrets with Alice and Bob! This is known as a man-in-the-middle attack. How can we prevent this?
- Certificates, which give a static public key verified by a trusted third party (Certificate Authority, or CA, whose public key is known) are a widely used solution.
- But, this requires trusting the CA's, which have been successfully attacked.

Problems with Public-Key Exchange

- Our protocol has a serious flaw.
- Mary can tamper with the Diffie-Hellman key exchange messages and establish her own shared secrets with Alice and Bob! This is known as a man-in-the-middle attack. How can we prevent this?
- Certificates, which give a static public key verified by a trusted third party (Certificate Authority, or CA, whose public key is known) are a widely used solution.
- But, this requires trusting the CA's, which have been successfully attacked.
- How can you prove your identity to another person over the internet in such a way that the other person can't later impersonate you?