Lecture 4

Gidon Rosalki

2025-11-19

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_intro_to_crypto`

1 Introduction

We are going to discuss Message Authentication Codes, where we verify the authenticity of a sent message. You receive a message, and can verify that it came from *me*, and nobody else (Like Eve. Seriously, screw Eve. Or don't, that might be what she wants).

We are going to discuss authenticating fixed length messages, and arbitrary length messages. We will do this through collision resistant hash functions, where we use the "hash and authenticate" paradigm. After this, we will link it back to encryption.

2 Message authentication

Alice and Bob wish to communicate, and Eve completely controls the channel. We would like to assure the receiver of a message, that it has not been modified. If Alice sends Bob a message that says "Pay Charlie \$10", Eve can change it to say "Pay Eve \$10,000" (You may also pretend that Bob is your Bank). Encryption ensures data secrecy, that no one else knows the contents, and *authentication* ensures the integrity of the data, that it remained unchanged. These concepts are orthogonal, one does not enable the other.

2.1 Message Authentication Code (MAC)

The syntax is $\Pi = (Gen, Mac, \text{Vrfy})$, where the key-generation algorithm Gen on input 1^n outputs a key k, the tag generation algorithm Mac takes a key k, and a message $m \in \{0,1\}^*$, and outputs a tag $t \in \{0,1\}^*$, and the verification algorithm Vrfy takes a key k, message m, and tag t, and outputs a bit b.
The correctness comes from:

$$\forall k, m \ \text{Vrfy}_k\left(m, Mac_k\left(m\right)\right) = 1$$

The security of MACs is found as follows: The adversary $\mathcal{A}$ can adaptively ask for tags of messages of its choice, and attempts to forge a valid tag on a new message (m^*, t^*). Let Q be the set of all queries asked by $\mathcal{A}$. We then have

$$MacForge_{\Pi,\mathcal{A}}\left(n\right) = \begin{cases} 1, & \text{if } \text{Vrfy}_k\left(m^*, t^*\right) = 1 \wedge m^* \notin Q \\ 0, & \text{otherwise} \end{cases}$$

Definition 2.1 (MAC scheme)**.** *A MAC scheme $\Pi = (Gen, Mac, Vrfy)$ is secure if for every PPT adversary $\mathcal{A}$, there exists a negligible function $v\left(\cdot\right)$ such that*

$$\Pr\left[MacForge_{\Pi,\mathcal{A}}\left(n\right) = 1\right] \leq v\left(n\right)$$

This definition does **not** prevent replay attacks. Eve may send the message "Send Charlie \$10" as many times as she likes, and it will appear legitimate, thanks to the deterministic MAC.

2.2 Fixed length MAC

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a PRF.

- Key generation: Sample $k \leftarrow \{0,1\}^n$
- Tag generation: On input $k \in \{0,1\}^n$ and $m \in \{0,1\}^n$ output $t = F_k\left(m\right)$
- Verification: On input $k \in \{0,1\}^n$, $m \in \{0,1\}^n$, and $t \in \{0,1\}^n$, output 1 if $t = F_k\left(m\right)$, and 0 otherwise

Theorem 1. *If F is a PRF, then the above MAC scheme is secure*

Proof . The concept is that given a forger $\mathcal{A}$, for the MAC scheme, we construct a distinguisher $\mathcal{D}$ for the PRF. $\mathcal{D}$ has oracle access to a function $\mathcal{O}$, which is either F_k, or a truly random h. Additionally, $\mathcal{D}$ runs $\mathcal{A}$ internally, and simulates the experiment $MacForge_{\Pi,\mathcal{A}}$ to $\mathcal{A}$ using $\mathcal{O}$.

Let us assume towards a contradiction that there exists a PPT adversary $\mathcal{A}$, and a polynomial $p(n)$ such that

$$\Pr\left[MacForge_{\Pi,\mathcal{A}}(n)=1\right]\geq\frac{1}{p(n)}$$

for infinitely many ns. The distinguisher $\mathcal{D}^{\mathcal{O}}$ will invoke $\mathcal{A}$, and respond to each of its queries m with $t=\mathcal{O}(m)$. It will output 1 **if and only if** $m^*\notin\mathcal{Q}$, and $t^*=\mathcal{O}(m^*)$, where $\mathcal{Q}$ is the set of all queries asked by $\mathcal{A}$. There are 2 cases:

Case 1: If $\mathcal{O}=F_k$ is a PRF, then $\mathcal{A}$'s view is identical to $MacForge_{\Pi,\mathcal{A}}(n)$, and so

$$\Pr\left[D^{F_k(\cdot)}(1^n)=1\right]=\Pr\left[MacForge_{\Pi,\mathcal{A}}(n)=1\right]$$

Case 2: If $\mathcal{O}=h$ is a truly random function, then if $m^*\notin\mathcal{Q}$, then $\mathcal{A}$'s view is independent of $\mathcal{O}(m^*)$, and so

$$\Pr\left[D^{h(\cdot)}(1^n)=1\right]=2^{-n}$$

From here, we may calculate

$$\begin{aligned}&\left|\Pr\left[D^{F_k(\cdot)}(1^n)=1\right]-\Pr\left[D^{h(\cdot)}(1^n)=1\right]\right| &(1)\\ &=\left|\Pr\left[MacForge_{\Pi,\mathcal{A}}(n)=1\right]-2^{-n}\right| &(2)\\ &\geq\frac{1}{p(n)}-2^{-n} &(3)\\ & &(4)\end{aligned}$$

So, we have successfully distinguished between a truly random function, and a PRF, which is a contradiction. □

2.3 Arbitrary length messages

Given

$$\widehat{\Pi}=\left(\hat{\text{Gen}},\hat{\text{Mac}},\hat{\text{Vrfy}}\right)$$

for fixed length messages, we want to define $\Pi=(Gen,Mac,\text{Vrfy})$ for arbitrary length messages. Here is a first (naïve) attempt:

2.3.1 Attempt 1

Let $Mac_k(m)=(t_1,\ldots,t_d)$ where $t_i=\widehat{\text{Mac}}_k(m_i)$, $Gen=\widehat{\text{Gen}}$, and $\text{Vrfy}_k((m_1,\ldots,m_d),(t_1,\ldots,t_d))=1$ **if and only if** $\widehat{\text{Vrfy}}_k(m_i,t_i)=1$ for every $i\in[d]$.

This is completely insecure. Consider

$$t=(t_1,t_2)$$

If t is avalid tag for $m=(m_1,m_2)$, then it also holds that $t^*=t_1$ is a valid tag for $m^*=m_1$.

2.3.2 Attempt 2

$Mac_k(m)=(t_1,\ldots,t_d)$, where $t_i=\widehat{\text{Mac}}_k(d,m_i)$, $\text{Gen}=\widehat{\text{Gen}}$, and $\text{Vrfy}_k((m_1,\ldots,m_d),(t_1,\ldots,t_d))=1$ **if and only if** $\widehat{\text{Vrfy}}_k((d,m_i),t_i)=1$ for every $i\in[d]$.

This is also insecure. Consider if $t=(t_1,t_2)$ is a valid tag for $m=(m_1,m_2)$, then $t^*=(t_2,t_1)$ is a valid tag for $m^*=(m_2,m_1)$.

2.3.3 Attempt 3

$Mac_k(m)=(t_1,\ldots,t_d)$, where $t_i=\widehat{\text{Mac}}_k(d,i,m_i)$, $\text{Gen}=\widehat{\text{Gen}}$, and $\text{Vrfy}_k((m_1,\ldots,m_d),(t_1,\ldots,t_d))=1$ **if and only if** $\widehat{\text{Vrfy}}_k((d,i,m_i),t_i)=1$ for every $i\in[d]$.

This is still insecure. If $t=(t_1,t_2)$ is a valid tag for $m=(m_1,m_2)$, and similarly $t'=(t'_1,t'_2)$ is a valid tag for $m'=(m'_1,m'_2)$, then $t^*=(t_1,t'_2)$ is a valid tag for $m^*=(m_1,m'_2)$

2.3.4 Solution 1

$Mac_k(m)=(r,t_1,\ldots,t_d)$, where $t_i=\widehat{\text{Mac}}_k(r,d,i,m_i)$, and r is sampled uniformly, and independently for each message m. $\text{Gen}=\widehat{\text{Gen}}$, and $\text{Vrfy}_k((m_1,\ldots,m_d),(r,t_1,\ldots,t_d))=1$ **if and only if** $\widehat{\text{Vrfy}}_k((r,d,i,m_i),t_i)=1$ for every $i\in[d]$.

This is a solution, but has the drawback of very long tags.

2.3.5 Solution 2 (CBC-MAC)

Let $Mac_k(m) = t_d$, where

- $t_0 = 0^n$, and $t_i = F_k(t_{i-1} \oplus m_i)$ for $i \in [d]$
- F_k can be any PRF
- d must be fixed ahead of time

In short, we take the result of the signature of the first block m_1, and XOR it with the second block m_2, and then compute the signature of that. This is continued on recursively until the end:

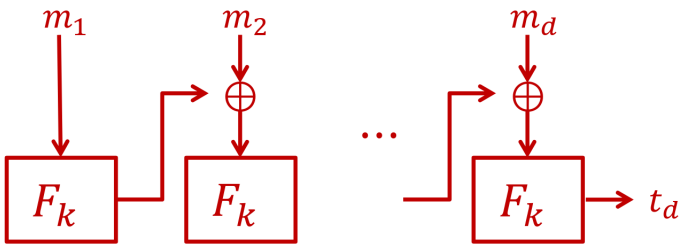

Figure 1: CBC-MAC

This is great because it is very easy to implement, so that solves many bug issues, and lots of the parts may be implemented in hardware.

2.3.6 Solution 3 - Hash and Authenticate

We will compress m into a short fingerprint $H(m)$, and then authenticate $H(m)$ instead of m itself. It is critical for our hashing function H, that it is very difficult to find $m \neq m'$ such that $H(m) = H(m')$.

3 Collision-Resistant Hash Functions

The purpose of Collision-Resistant Hash Functions is that they compress arbitrarily long inputs into short, fixed-length outputs. It should be very hard to find $x \neq x'$ such that $H(x) = H(x')$.

Syntactically: $\Phi = (\text{Gen}, H)$:

- The key generation algorithm Gen, on input 1^n outputs a key s
- The evaluation algorithm H on input s, and $x \in \{0,1\}^*$ outputs $H_s(x) \in \{0,1\}^{l(n)}$

So, our adversary is playing

$$HashColl_{\Phi,\mathcal{A}}(n) = \begin{cases} 1, & \text{if } H_s(x) = H_s(x') \wedge x \neq x' \\ 0, & \text{otherwise} \end{cases}$$

Bringing us to the definition

Definition 3.1 (Collision Resistant)**.** Φ *is* ***collision resistant*** *if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $v(\cdot)$ *such that*

$$\Pr[HashColl_{\Phi,\mathcal{A}}(n) = 1] \leq v(n)$$

Can we find collisions? Well, if our function produces output of length n bits, then we could $2^n + 1$ inputs, but this will take forever. Instead, we have **The Birthday Attack**.

Given $H : \{0,1\}^* \to \{0,1\}^l$, sample $q = \mathcal{O}\left(2^{\frac{l}{2}}\right)$ inputs uniformly and independently. This finds colliding pair of inputs with a constant probability (2^l pairs). To think about this in general, given output of 2^n, then similar to the birthday paradox from probability and statistics, taking $\sqrt{2^n} = 2^{\frac{n}{2}}$ possible inputs, and testing them, will give us a shared output with probability of 50%.

In practice $l \geq 128$. Popular heuristic (unkeyed) functions include MD5, SHA1, SHA3, and so on. Both MD5, and SHA1 are horrifically insecure, but still heavily used. They can still be useful in non crypto contexts, but should now *never* be used in a crypto context.

Generally, in order to to create an arbitrary length hash function, we start with a fixed length has function, and extend it.

4 Authenticating Arbitrary-Length Messages

Given

$$\widehat{\Pi} = \left(\hat{\text{Gen}}, \hat{\text{Mac}}, \hat{\text{Vrfy}}\right)$$

for fixed length messages, and a collision resistant hash function $\Phi = (Gen_H, H)$, define $\Pi = (Gen, Mac, \text{Vrfy})$ for arbitrary length messages.

Reconsider Solution 3 above:

- $Mac_{k,s}(m) = \widehat{Mac}_k(H_s(m))$
- $Gen = \left(\widehat{Gen}, Gen_H\right)$
- $\text{Vrfy}_{k,s}(m,t) = 1 \Leftrightarrow \widehat{Vrfy}_k(H_s(m), t) = 1$

Exercise 1. *Let $Mac_k(m)$ samples $s \leftarrow Gen_H(1^n)$, and outputs $\left(s, \widehat{Mac}_k(H_s(m))\right)$. Is this secure?*

Solution. This is insecure. In general, our family our functions should overall be difficult to find collisions within, but it is *possible* that there exists a function within the family where it is incredibly easy to find within it collisions. Since s is generated, our adversary can choose an s for which the function H_s has many collisions. Consider the following:

- The adversary requests the signature of 0, and gets in response $(s, Mac_k(H_s(0)))$. Let us denote $a = H_s(0)$.
- We will now find a function within the family that maps *everything* to a.
- The adversary will now pick this value for s, and the signature of every message m will be

$$Mac_k(H_s(m)) = Mac_k(a) = Mac_k(H_s(0))$$

□

Let us return to Solution 3.

Theorem 2. *If $\widehat{\Pi}$ is a secure MAC, and Φ is collision resistant, then Π is a secure MAC.*

Proof . Consider the event "collision". $\mathcal{A}$ asks for a tag on some m_i such that $m_i \neq m^*$, and $H_s(m_i) = H)s(m^*)$. Whenever "collision" occurs, then we can use $\mathcal{A}$ to find a non trivial collision, and whenever "collision" does not occur (and $\mathcal{A}$ wins), then we can use $\mathcal{A}$ to forge a tag on the fixed length message

$$H_s(m^*) \notin \{H_s(m_1), \ldots, H_s(m_q)\}$$

Let $\mathcal{A}$ be any PPT adversary, then

$$\Pr[MacForge_{\Pi,\mathcal{A}}(n) = 1] \leq \Pr[\text{collision}] + \Pr\left[MacForge_{\Pi,\mathcal{A}}(n) \wedge \overline{\text{collision}}\right]$$

Since

$$\Pr[A] = \Pr[A \wedge B] + \Pr\left[A \wedge \overline{B}\right] \leq \Pr[B] + \Pr\left[A \wedge \overline{B}\right]$$

Let **claim I** be that there exists a negligible function $v_1(n)$ such that

$$\Pr[\text{collision}] \leq v_1(n)$$

and **claim II** be that there exists a negligible function $v_2(n)$ such that

$$\Pr\left[MacForge_{\Pi,\mathcal{A}}(n) \wedge \overline{\text{collision}}\right] \leq v_2(n)$$

Proof of Claim I:
Let us assume towards a contradiction that there exists a PPT adversary $\mathcal{A}$, and a polynomial $p(n)$ such that

$$\Pr[\text{collision}] \geq \frac{1}{p(n)}$$

The collision finder $\mathcal{C}$ will

- On input s, sample $k \leftarrow \widehat{Gen}(1^n)$, and invoke $\mathcal{A}$
- Respond to $\mathcal{A}$'s queries, using (k, s)

Proof of Claim II:
Let us assume towradrs contradiction that there exists a PPT adversary $\mathcal{A}$, and a polynomial $p(n)$, such that

$$\Pr\left[MacForge_{\Phi,\mathcal{A}}(n) = 1 \wedge \overline{\text{collision}}\right] \geq \frac{1}{p(n)}$$

for infinitely many ns. The forger $\hat{\mathcal{A}}$ will

- Sample $s \leftarrow Gen_H(1^n)$, and invoke $\mathcal{A}$
- Respond to $\mathcal{A}$'s queries m_i by forwarding $H_s(m_i)$ to the oracle $\widehat{Mac}_k(\cdot)$, and then forwarding back its response to $\mathcal{A}$
- Output $(H_s(m^*), t^*)$

So

$$\begin{aligned}\Pr\left[MacForge_{\hat{\Pi},\hat{\mathcal{A}}}(n) = 1\right] &= \Pr\left[MacForge_{\Pi,\mathcal{A}}(n) = 1 \wedge \overline{\text{collision}}\right] \\ &\geq \frac{1}{p(n)}\end{aligned}$$

Which is a contradiction. □

5 Returning to encryption

Recall that CPA-secure encryptions may be built from any PRF, and hold that

$$Enc_k(m; r) = (r, F_k(r) \oplus m)$$

This has a rather serious problem. An adversary can change the contents of a message, without the recipient knowing. The adversary will also not know how he has changed it, but he can change it, and none will know that this has happened.

$$Enc_k(m \oplus 1^n; r) = (r, F_k(r) \oplus m \oplus 1^n)$$

So, we want to achieve both encryption, and authenticated messages, creating **authenticated encryption**.
Let us consider

$$k = (k_E, k_M) \tag{5}$$
$$c \leftarrow Enc_{k_E}(m) \tag{6}$$
$$t \leftarrow Mac_{k_M}(m, t) \tag{7}$$

Which can then all be reversed

$$k = (k_E, k_M) \tag{8}$$
$$m \leftarrow Dec_{k_E}(c) \tag{9}$$
$$? \leftarrow Mac_{k_M}(m, t) \tag{10}$$

This may well be insecure. We have no guarantees that our MAC system does not leak information about the message. In fact, we may completely leak m, since we have no security guarantees.
This may be improved to

$$k = (k_E, k_M) \tag{11}$$
$$c \leftarrow Enc_{k_E}(m) \tag{12}$$
$$t \leftarrow Mac_{k_M}(c, t) \tag{13}$$

Which can then all be reversed

$$k = (k_E, k_M) \tag{14}$$
$$m \leftarrow Dec_{k_E}(c) \tag{15}$$
$$? \leftarrow Mac_{k_M}(c, t) \tag{16}$$

If the encryption is CPA-secure, and the MAC is secure, then this construction is a CPA-secure encryption scheme, with a secure MAC.

5.1 Chosen Ciphertext Attack CCA

This has brought us to chosen ciphertext attack schemes. Here, $\mathcal{A}$ can adaptively ask for encryptions of messages of its choice, and for decryptions of ciphertexts of its choice (aside from the challenge ciphertext c^*).

Definition 5.1 (CCA-IND)**.** Π *has indistinguishable encryptions under a chosen-ciphertext attack if for every PPT adversary* $\mathcal{A}$ *there exists a negligible function* $v(\cdot)$ *such that*

$$\Pr\left[IND^{CCA}_{\Pi,\mathcal{A}}(n)=1\right] \leq \frac{1}{2}+v(n)$$

In this case, we may also say that Π *is CCA-secure.*

We will observe that CCA-security implies authenticity. Given $Enc_k(m)$, it is hard to generate $Enc_k(m')$ for a "related" m' (such as $m'=m+1$). This is because if he could, then he could take c^*, and flip the bits. This is (under the assumption) a legal encryption, which could then be decrypted into the inverse bits of m_b, so he could then take the decryption of the inverse of c^*, flip its bits, and identify which m_i was returned encrypted to c^*.

We will note that it feels a little unrealistic. Honest parties do not typically decrypt arbitrary adversarially chosen ciphertexts. Nevertheless, adversaries may be able to influence what gets encrypted / decrypted, and learn some partial information. For example, in WWII, the US cryptanalysts might have tried to send encrypted messages to the Japananese, and then monitor their behaviour. An adversary may send certain ciphertexts on behalf of a user to the user's bank, and then the bank will decrypt these ciphertexts and its response will leak information to the adversary. Furthermore, an encryption scheme might be used as part of an authentication protocol where one party sends a ciphertext to the other, who then decrypts it and returns the result. Thus we can see, that CCA is a realistic security requirement in the real world.

We will thus note that CPA security does **not** imply CCA security, since we can affect the output of encryption function, and get another valid encryption:

$$Enc_k(m;r) = (r, F_k(r)\oplus m)$$
$$Enc_k(m\oplus 1^n;r) = (r, F_k(r)\oplus m\oplus 1^n)$$

5.1.1 CCA-Secure encryption scheme

The main idea here is that adversaries should not be able to generate new "valid" ciphertexts, and that the decryption oracle becomes useless. Our solution is called *Encrypt-then-Authenticate*:
Let $\Pi_E=(KeyGen_E, Enc, Dec)$ be a CPA secure encryption scheeme, and let $\Pi_M=(KeyGen_M, Enc, Dec)$ be a secure MAC. Let us define $\Pi'=(KeyGen', Enc', Dec')$ as

- $KeyGen'(1^n)$ output $k=(k_E,k_M)$ where $k_E \leftarrow Gen_E(1^n)$ and $k_M \leftarrow KeyGen_M(1^n)$
- $Enc'_k(m)$ output (c,t) where $c \leftarrow Enc_{k_E}(m)$ and $t \leftarrow Mac_{k_M}(c)$
- $Dec'_k(c,t)$ If $Vrfy_{k_M}(c,t)=1$ then output $Dec_{k_E}(c)$, otherwise output $\perp$

Theorem 3. *Let us assume that* Π_E *is a CPA-secure encryption scheme, and that* Π_M *is a secure MAC with* ***unique tags****, then* Π' *is a CCA secure encryption scheme.*

Proof . We will first note that MAC with unique tags means that for any key k_M and message m, there exists *exactly* one t such that $Vrfy_{k_M}(m,t)=1$. Any MAC scheme with a deterministic Mac algorithm can be modified to have unique tags by using "canonical verification" instead of its own Vrfy algorithm: On input (k_M,m,t) output 1 if and only if $t=Mac_{k_M}(m)$ and 0 otherwise

Proof idea:
We will call (c,t) valid w.r.t (k_E,k_M) if $Vrfy_{k_M}(c,t)=1$, and invalid otherwise. Consider the event Valid-Query. $\mathcal{A}$ queries the decryption oracle with a valid ciphertext, that was **not** produced by the encryption oracle. If $\Pr[ValidQuery]$ is non-negligible, then we can use $\mathcal{A}$ to break the MAC Π_M. If it is negligible, then we can use $\mathcal{A}$ to break the CPA security of Π_E

Let $\mathcal{A}$ be a PPT adversary, then:

$$\Pr\left[IND^{CCA}_{\Pi',\mathcal{A}}(n)=1\right] \tag{17}$$
$$\leq \Pr[ValidQuery] + \Pr\left[IND^{CCA}_{\Pi',\mathcal{A}}(n)=1 \wedge \overline{ValidQuery}\right] \tag{18}$$

Theorem 4 (Claim II)**.** ***Claim II****: There exists a negligible* $v(n)$ *such that*

$$\Pr\left[IND^{CCA}_{\Pi',\mathcal{A}}(n)=1 \wedge \overline{ValidQuery}\right] \leq \frac{1}{2}+v(n)$$

Proof . We will assume towards contradiction that there exists a polynomial $p(n)$, such that for infinitely many ns

$$\Pr\left[IND^{CCA}_{\Pi',\mathcal{A}}(n)=1 \wedge \overline{ValidQuery}\right] \geq \frac{1}{2}+\frac{1}{p(n)}$$

We will construct an adversary $\mathcal{B}$ such that for infinitely many ns

$$\Pr\left[IND^{CCA}_{\Pi_E,\mathcal{B}}(n) = 1 \wedge \overline{ValidQuery}\right] \geq \frac{1}{2} + \frac{1}{p(n)}$$

$\mathcal{B}$ will sample k_M, and invoke $\mathcal{A}$, and respond to its queries. It will then output the same (m_0, m_1), and b'.
It will query the oracle $Enc_{k_E}(\cdot)$ to obtain a ciphertext c, compute $t \leftarrow Mac_{k_M}(c)$, and return (c,t).
For the decryption query, if (c,t) was a response to a previous query m, then return m, and otherwise return $\perp$.

Since ValidQuery does not occur, then $\mathcal{B}$ does not need a decryption oracle for simulating $\mathcal{A}$'s view. Thus:

$$\begin{aligned}
&\Pr\left[IND^{CPA}_{\Pi_E,\mathcal{B}}(n) = 1\right] && (19)\\
&\geq \Pr\left[IND^{CPA}_{\Pi_E,\mathcal{B}}(n) \wedge \overline{ValidQuery}\right] && (20)\\
&= \Pr\left[IND^{CPA}_{\Pi_E,\mathcal{A}}(n) \wedge \overline{ValidQuery}\right] && (21)\\
&\geq \frac{1}{2} + \frac{1}{p(n)} && (22)
\end{aligned}$$

□

Theorem 5 (Claim I)**.** ***Claim I****: There exists a negligible $v(n)$ such that $\Pr[ValidQuery] \leq v(n)$*

Proof . Let us assume towards contradiction that there exists a polynomial $p(n)$ such that $\Pr[ValidQuery] \geq \frac{1}{p(n)}$ for infinitely many n's. Let $q(n)$ be a polynomial upper bound on the number of decryption queries made by $\mathcal{A}$. We will construct a PPT adversary $\mathcal{B}$ such that $\Pr[MacForge_{\Pi_M,\mathcal{B}} = 1] \geq \frac{1}{p(n)\cdot q(n)}$ for infinitely many ns

We will construct $\mathcal{B}$ as follows: Sample k_E, b and $i \leftarrow \{1,\ldots,q\}$, invoke $\mathcal{A}$, and respond to its queries. For an encryption query m, we will compute $c \leftarrow Enc_{k_E}(m)$, and query the oracle $Mac_{k_M}(\cdot)$ with c to obtain the tag t. Finally, return (c,t).
For the first $i-1$ decryption queries (c,t), if (c,t) was a response to a previous encryption query m, then return m, and otherwise $\perp$. For the ith decryption query (c,t), output $(c^*,t^*) = (c,t)$ as the (potential) forgery.

We will observe that ValidQuery occurs, and that $\mathcal{B}$ geusses the index i of the first valid decryption query that was not obtained from a previous encryption query. Therefore, $Vrfy_{k_M}(c^*,t^*) = 1$ and $\mathcal{B}$ did not query $Mac_{k_M}(\cdot)$ on c^*. So, we may conclude

$$\begin{aligned}
\Pr[MacForge_{\Pi_M,\mathcal{B}} = 1] &\geq \Pr[ValidQuery \wedge \mathcal{B} \text{ guesses } i]\\
&= \frac{\Pr[ValidQuery]}{q(n)}\\
&\geq \frac{1}{p(n)\cdot q(n)}
\end{aligned}$$

□

□

6 Crypto primitives so far

So far, we have seen PRFs, PRGs, built PRGs from PRFs. From PRGs we have create IND-secure symmetric key encryption, and built CPA-secure symmetric key encryption from PRFs. Furthermore, we built IND-secure from CPA-secure. PRFs were also used to create MACs for fixed length messages, we then combined fixed length MAC and CPA-secure to make CCA secure symmetric key encryption. Finally, from MAC for fixed length messages, and Collision Resistant Hash Function, we made MAC for arbitrary length messages.

This concludes this section of the course, covering symmetric encryption.