1)

الف)

توابع درهم‌سازی برای تبدیل داده‌ها (معمولاً با طول متغیر) به یک مقدار ثابت طول به نام هش استفاده می‌شوند. هدف اصلی استفاده از توابع درهم‌سازی شامل مواردی مانند یکسان‌سازی داده‌ها، تولید شناسه یکتا و ثابت برای داده‌های مختلف، تضمین صحت و تمامیت داده‌ها، افزایش امنیت و جلوگیری از تصادف است. با این توابع می‌توان از تغییرات ناخواسته در داده‌ها جلوگیری کرد و امنیت اطلاعات حساس را افزایش داد.

ب)

برای تضمین صحت محتوای فایل‌ها می‌توان ابتدا فایل را از طریق یک تابع درهم‌سازی مانند SHA-256 به یک مقدار هش ثابت تبدیل کرد و این مقدار هش را به عنوان امضای دیجیتال فایل ذخیره یا ارسال کرد. سپس هر زمان که نیاز به بررسی صحت فایل بود، مقدار هش جدید فایل محاسبه و با مقدار هش قبلی مقایسه می‌شود. اگر مقدار هش جدید با مقدار هش قبلی یکسان باشد، به این معنی است که فایل تغییر نکرده است و در غیر این صورت نشان‌دهنده تغییر یا دستکاری در محتوای فایل است.

ج)

خیر، این گزاره لزوماً درست نیست. مقاومت در برابر تصادم به این معنی است که پیدا کردن دو پیام متفاوت که هش یکسانی داشته باشند بسیار دشوار است، اما غیرممکن نیست. به دلیل اینکه تابع درهم‌سازی یک مقدار با طول ثابت تولید می‌کند و پیام‌ها می‌توانند طول دلخواهی داشته باشند، طبق اصل لانه کبوتری تعداد محدودی مقدار هش برای تعداد نامحدودی ورودی وجود دارد. بنابراین، همواره امکان وجود تصادف وجود دارد، یعنی دو پیام متفاوت که مقدار هش یکسانی تولید کنند.

2)

الف)

امضای دیجیتال به منظور تضمین **یکپارچگی** و **احراز هویت** در ارتباطات دیجیتالی طراحی شده است. این سیستم بر مبنای استفاده از **کلیدهای عمومی و خصوصی** عمل می‌کند که در رمزنگاری نامتقارن کاربرد دارند. برای ایجاد یک امضای دیجیتال، فرستنده با استفاده از **کلید خصوصی** خود، یک هش از پیام را رمزگذاری کرده و به عنوان امضا همراه با پیام ارسال می‌کند. در طرف دیگر، گیرنده با استفاده از **کلید عمومی فرستنده**، امضا را رمزگشایی کرده و هش پیام را بازیابی می‌کند. سپس با محاسبه هش از پیام اصلی، صحت امضا و محتوا را بررسی می‌کند. امنیت امضای دیجیتال به **محرمانه بودن کلید خصوصی** و **قابلیت تأیید امضا با کلید عمومی** بستگی دارد.

ب)

الگوریتم **RSA**از کلید خصوصی برای رمزگذاری پیام به عنوان امضا استفاده می‌کند و گیرنده با استفاده از کلید عمومی، امضا را رمزگشایی کرده و آن را تأیید می‌کند. این الگوریتم هم برای **رمزنگاری** و هم **ایجاد امضا** به کار می‌رود. اما در **DSA**، امضا با استفاده از کلید خصوصی به همراه یک مقدار تصادفی ایجاد می‌شود و کلید عمومی برای تأیید استفاده می‌شود DSA .تنها برای امضاهای دیجیتال طراحی شده است و برای رمزنگاری کاربرد ندارد. در مجموع، RSAبه دلیل عملیات ساده‌تر، سرعت بیشتری در تولید امضا دارد، اما DSA با استفاده از مقدار تصادفی، امنیت بیشتری در برابر حملات تکرار فراهم می‌کند.

ج)

در امضاهای دیجیتال، کلیدهای عمومی و خصوصی نقش مهمی دارند .**کلید خصوصی** برای ایجاد امضا استفاده می‌شود و باید به صورت محرمانه نزد صاحب امضا باقی بماند. در هر دو الگوریتم DSA وRSA ، کلید خصوصی نباید فاش شود و فقط برای صاحب امضا شناخته شده است .**کلید عمومی** برای تأیید امضا به کار می‌رود و می‌تواند به صورت عمومی توزیع شود، که به گیرندگان امکان می‌دهد بدون دسترسی به کلید خصوصی، صحت امضا را بررسی کنند.

مزایا و پیامدهای استفاده از کلیدهای عمومی و خصوصی شامل موارد زیر است

تأیید امضا به راحتی امکان‌پذیر است، بدون اینکه نیازی به کلید خصوصی باشد

امنیت سیستم افزایش یافته و خطر جعل امضا و حملات تکرار کاهش می‌یابد

امکان احراز هویت دیجیتال و ایجاد اعتماد بین طرفین فراهم می‌شود، زیرا هر شخص می‌تواند هویت خود را با کلید خصوصی‌اش به اثبات برساند

(3

سازگاری و استانداردسازی HMAC بر پایه الگوریتم‌های هش‌گذاری مانند SHA-256 طراحی شده و به طور گسترده در پروتکل‌های مختلف استانداردسازی شده است. این موضوع سبب می‌شود که استفاده از آن در سیستم‌های مختلف سازگاری بیشتری داشته باشد

سهولت در پیاده‌سازی و استفاده گسترده از آنجا که HMAC بر اساس الگوریتم‌های هش‌گذاری است، پیاده‌سازی آن ساده‌تر است و در بیشتر کتابخانه‌های امنیتی موجود است. در مقابل، CMACکه بر پایه رمزنگاری متقارن مانند AES است، نیاز به الگوریتم‌های رمزنگاری بلوکی دارد که ممکن است در همه پروتکل‌ها و کتابخانه‌ها پشتیبانی نشود

پایداری در عملکرد HMAC در شرایط مختلف کارایی و امنیت بالایی را نشان داده است و به خوبی در برابر حملات مقاومت می‌کند، بنابراین در پروتکل‌های مهم و حیاتی مثل TLSترجیح داده می‌شود

در برخی شرایط CMAC می‌تواند گزینه بهتری باشد

زمانی که امنیت بیشتری در برابر حملات خاص مانند حملات طولی مورد نیاز است، CMACمی‌تواند انتخاب بهتری باشد زیرا بر پایه رمزنگاری بلوکی است

در محیطهایی که از AES به عنوان استاندارد رمزنگاری استفاده می‌شود، CMAC می‌تواند امنیتی هم‌سطح با AES ارائه دهد و به این ترتیب امنیت کلی سیستم افزایش می‌یابد

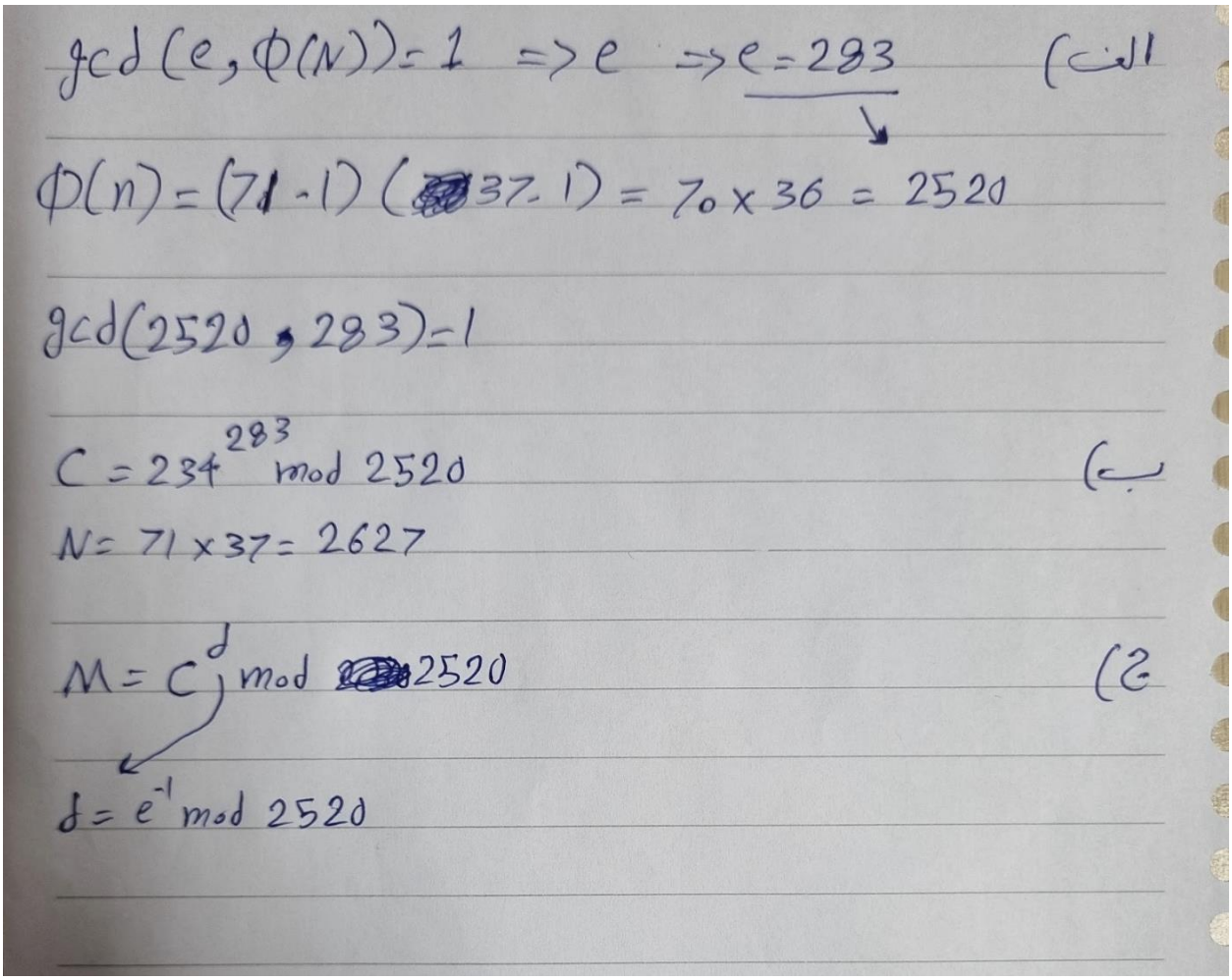
استفاده از کلیدهای قوی و امن برای AES یا الگوریتم بلوکی دیگر که CMAC بر پایه آن کار می‌کند

مدیریت و ذخیره‌سازی امن کلیدها به دلیل وابستگی CMAC به کلیدهای رمزنگاری

استفاده از پروتکل‌های مقاوم در برابر حملات تصادف و حملات زمان‌بندی که ممکن است در رمزنگاری بلوکی وجود داشته باشد

در نتیجه، HMACبه دلیل سهولت در پیاده‌سازی، سازگاری بالا و پشتیبانی گسترده در پروتکل‌های امنیتی مانند TLS بیشتر مورد استفاده قرار می‌گیرد اما در شرایطی که نیاز به امنیت بالاتر و استفاده از الگوریتم‌های بلوکی مانند AES باشد، CMACمی‌تواند انتخاب مناسبی باشد

(4



(5

جایگزینی کلید :مهاجم (که معمولاً او را **مالوری** می‌نامند) در حین تبادل کلید بین آلیس و باب وارد ارتباط می‌شود. آلیس پیام کلید عمومی خود را به باب ارسال می‌کند، اما مالوری این پیام را شنود و از راه تغییر، پیام خود را به جای پیام آلیس به باب می‌فرستد.

تبادل کلید مستقل :مالوری کلید خود را با آلیس و باب تبادل می‌کند. به این ترتیب، مالوری دو ارتباط مستقل ایجاد کرده است: یکی با آلیس و دیگری با باب.

شنود و تغییر پیام‌ها :هر پیامی که آلیس برای باب ارسال می‌کند، توسط مالوری دریافت شده و ممکن است قبل از ارسال به باب تغییر داده شود و بالعکس. بنابراین، مالوری به عنوان واسطه قادر است پیام‌ها را بخواند یا حتی دستکاری کند.