

امنیت

اسلاید اول:

🔑 امنیت = حفاظت از چیزهای مهم

🔑 امنیت کامپیوتر = حفاظت از دارایی‌های سیستم/کامپیوتر در برابر آسیب، از دست رفتن یا دسترسی غیرمجاز

🔑 آسیب‌پذیری: (Vulnerability)

- یک ضعف در سیستم (در روش‌ها، طراحی یا پیاده‌سازی)
- به‌تنهایی آسیبی نمی‌زند، اما زمینه را برای سوءاستفاده تهدید فراهم می‌کند

🔑 تهدید: (Threat)

- رویداد یا عملی که می‌تواند از یک آسیب‌پذیری سوءاستفاده کند و آسیب یا خسارت ایجاد کند.
- می‌تواند عمدی یا غیرعمدی باشد.

🔑 راه‌حل مشکلات امنیتی:

- استفاده از کنترل/راهکار (Control/Countermeasure) برای حفاظت.
- کنترل = اقدام، وسیله، روش یا تکنیکی برای حذف یا کاهش آسیب‌پذیری.
- تهدید با کنترل آسیب‌پذیری مسدود می‌شود.

🔑 خواص اصلی امنیت:

- محرمانگی: (Confidentiality) فقط افراد مجاز بتوانند دارایی را ببینند.
- یکپارچگی: (Integrity) فقط افراد مجاز بتوانند دارایی را تغییر دهند.
- دسترسی‌پذیری: (Availability) دارایی همیشه برای افراد مجاز در دسترس باشد.
- احراز هویت: (Authentication) تأیید هویت فرستنده.
- انکارناپذیری / پاسخگویی: (Non-repudiation/Accountability) فرستنده نتواند ارسال خود را انکار کند.

- تهدیدها می‌توانند انسانی یا غیرانسانی باشند.

- تمرکز اصلی: تهدیدهای انسانی
  - غیر مخرب (Non-malicious)
  - مخرب → (Malicious) حمله: (Attack) فرد مخرب قصد آسیب دارد.

- امنیت کامپیوتر: تمرکز روی دستگاه‌های منفرد
- دستگاه‌ها معمولاً متصل به شبکه هستند → مفهوم سایر
  - **Cyberthreat:** تهدید علیه شبکه‌ای از کامپیوترها
  - **Cyberspace:** فضای آنلاین (به‌ویژه اینترنت)
  - **Cybercrime:** حمله غیرقانونی به کامپیوترهای شبکه، کاربران، داده، سرویس‌ها و زیرساخت
- امنیت هم برای دستگاه منفرد و هم شبکه → امنیت سایبری (Cybersecurity)

- کنترل/پادکنش: راه مقابله با تهدیدها
- آسیب زمانی رخ می‌دهد که تهدید از ضعف سوءاستفاده کند
- روش‌های مقابله با آسیب:
  - **Prevent:** جلوگیری از حمله یا بستن ضعف
  - **Deter:** سخت‌تر کردن حمله بدون غیرممکن کردن آن
  - **Deflect:** هدایت حمله به هدف دیگر یا کاهش جذابیت هدف
  - **Mitigate:** کاهش شدت اثر
  - **Detect:** شناسایی هنگام وقوع یا بعداً
  - **Recover:** جبران اثرات

## اسلاید دوم

- ابزارهای کلیدی امنیت:
  - **Authentication:** تأیید هویت
  - **Access control:** کنترل دسترسی
  - **Cryptography:** رمزنگاری

بیس امنیت کامپیوتر = controlled access

- **Authentication:** فرآیند تأیید هویت برای اطمینان از اینکه فرد مجاز برای انجام یک عمل مشخص است.

- **Identification:** وقتی به نفر هویت خودش رو اعلام می‌کنه. هویت‌ها معمولاً شناخته شده یا قابل حدس هستن. تو خیلی کارا به بخشی از هویت رو نشون میدی و ممکنه کس دیگه‌ای با استفاده از اون بخش، خودش رو جای تو جا بزنه.
- **Authentication:** وقتی ثابت می‌کنیم اون هویتی که گفته شده درسته و شخص همون کسیه که ادعا می‌کنه. این کار باید خصوصی انجام بشه.

#### • Authentication Mechanisms:

1. چیزی که کاربر می‌دونه: مثل پسورد، PIN یا به روش محرمانه دیگه.
2. چیزی که کاربر هست: ویژگی‌های فیزیکی مثل اثر انگشت، صدا، چهره (بیومتریک).
3. چیزی که کاربر داره: کارت شناسایی، کلید فیزیکی، گواهینامه رانندگی.

#### • پسورد:

- روش نسبتاً امنی برای تأیید هویت.
- ولی امنیتش به رفتار آدم‌ها بستگی داره و گاهی خراب میشه.
- سؤال: پسوردها واقعاً چقدر امن هستن؟

#### • حملات دیکشنری و حدس پسورد:

- سایت‌ها دیکشنری از کلمات و اسم‌ها دارن که هکرها می‌تونن استفاده کنن.
- بعضی‌ها فکر می‌کنن با جایگزینی کاراکترها مثل 0 به جای O یا 3 به جای E باهوش شدن، ولی هکرها هم همینو بلد هستن.

#### • پسوردهای شخصی و قابل حدس:

- افراد معمولاً از اسم همسر، فرزند، خانواده یا حیوان خانگی برای پسورد استفاده می‌کنن چون راحت‌تر یادشون می‌مونه.
- تحقیقات نشون داده مردم خیلی وقت‌ها پسوردهای ضعیف و راحت حدس‌زدنی انتخاب می‌کنن.

#### • پسوردهای کوتاه:

- افراد معمولاً پسورد کوتاه رو ترجیح می‌دن.

- تعداد پسوردهای کوتاه (مثلاً طول  $\geq 3$ ) خیلی کمه و راحت با برنامه‌های تست پسورد حدس زده می‌شه.
- حتی با سرعت یک پسورد در هر میلی ثانیه، پسوردهای کوتاه در چند ثانیه شکسته می‌شن.
- **محدودیت لاگین:**
  - اکثر سیستم‌ها بعد از چند تلاش ناموفق، کاربر رو قفل می‌کنن.
  - این خودش می‌تونه وسیله‌ای برای حمله و جلوگیری از دسترسی کاربر واقعی باشه.
- خواندن مستقیم پسورد: راحت‌تر از حدس زدن پسورده، مخصوصاً از جایی که ذخیره شده.
- ذخیره امن: سیستم‌عامل‌ها پسوردها رو به صورت آشکار ذخیره نمی‌کنن، بلکه اون‌ها رو مخفی یا رمزنگاری شده نگه می‌دارن.
- احراز هویت: وقتی کاربر پسورد می‌ده، سیستم همون تابع مخفی‌کننده رو روی ورودی اعمال می‌کنه و با نسخه ذخیره‌شده مقایسه می‌کنه.
- فرآیند مخفی‌سازی یک‌طرفه است: از پسورد به نسخه مخفی راحته، برعکس تقریباً غیرممکن.
- تهدید در صورت لو رفتن جدول: اگر جدول رمزگذاری شده و الگوریتم مخفی‌سازی به دست مهاجم بیفته، می‌تونه صدها هزار حدس رو سریع امتحان کنه.
- استفاده از **Rainbow Table** لیست پیش‌محاسبه‌شده نسخه‌های مخفی پسوردهای رایج برای سرعت بخشیدن به حدس‌ها.
- پسوردهای مخفی‌شده هم مشکل دارند: اگر دو نفر همان پسورد را انتخاب کنند، نسخه مخفی‌شده‌شان یکسان خواهد بود.
- راه حل: اضافه کردن **salt** به فرآیند مخفی‌سازی پسورد.
- **Salt** یه داده اضافیه که برای هر کاربر متفاوت (مثل تاریخ ساخت حساب یا قسمتی از اسم کاربر).
- **Salt** به پسورد اضافه می‌شه قبل از مخفی‌سازی → پسوردها حتی اگر یکسان باشن، نسخه مخفی‌شده‌شون فرق می‌کنه.
- با این روش، جدول رنگین‌کمانی (rainbow table) دیگه کاربرد نداره چون هر پسورد یه بخش یکتا داره.

- **Brute force attack:** حمله کننده همه پسوردهای ممکن رو امتحان می کنه.
- تعداد پسوردهای ممکن بستگی به سیاست پسورد و سیستم داره.
- مثال: پسوردهای ۱ تا ۸ کاراکتری از حروف  $A-Z \rightarrow 5 \times 10^{12}$  پسورد ممکن.
- سرعت بررسی ۱ پسورد در میلی ثانیه  $\rightarrow$  حدود ۱۵۰ سال، ۱ پسورد در میکروثانیه  $\rightarrow$  حدود ۲ ماه.
- معمولاً نصف فضای پسورد کافی برای پیدا کردن یک پسورده.
- می تونیم زمان حمله رو طولانی تر کنیم با اجازه دادن به اعداد و کاراکترهای ویژه (#, \$, @) در پسورد.

- **روش های ساده برای امنیت پسورد:**
  - از اسامی واقعی یا کلمات معمولی استفاده نکنید
  - از حروف a-z تنها استفاده نکنید، کاراکترهای دیگر هم بیارید
  - پسوردهای بلند انتخاب کنید
  - پسورد را مرتب تغییر دهید
  - پسورد را ننویسید و به کسی نگویند
- **حمله آسان:** مهندسی اجتماعی، مثلاً تماس گرفتن و وانمود کردن به مدیر سیستم برای گرفتن پسورد.

- **احراز هویت مبتنی بر بیومتریک:**
  - ویژگی های فیزیکی یا بیولوژیکی بدن انسان برای شناسایی استفاده می شوند
  - نمونه ها: اثر انگشت، چهره، عنبیه چشم، صدا، دست خط/امضا، حرکت دست

- برای استفاده از بیومتریک ابتدا باید در سیستم ثبت نام کنید
  - سیستم قالب ویژگی های شما را ذخیره می کند
- هنگام احراز هویت با بیومتریک:
  - مجموعه ای از اندازه گیری های فعلی گرفته می شود و با قالب ذخیره شده مقایسه می شود
  - احراز هویت موفق است اگر تطابق «به اندازه کافی نزدیک» باشد

- مزایای بیومتریک نسبت به رمز عبور:
  - قابل گم شدن، سرقت یا فراموش شدن نیست
  - همیشه در دسترس است

- برخی ویژگی‌ها ممکن است جعل شوند اما سختی انجام آن بالا است

- دو نوع خطا در بیومتریک:

- **False positive:** اشتباهاً هویت را تأیید می‌کند
- **False negative:** اشتباهاً هویت را رد می‌کند

- **Sensitivity یا recall, نرخ درست مثبت:**

- توانایی سیستم در شناسایی درست هویت‌های واقعی مثبت
- هرچه حساسیت بالاتر باشد، تعداد **false negative** کمتر است
- فرمول:

$$\text{Sensitivity} = \text{TP} / (\text{TP} + \text{FN})$$

- **Specificity نرخ درست منفی:**

- توانایی سیستم در شناسایی درست هویت‌های واقعی منفی
- هرچه specificity بالاتر باشد، تعداد **false positive** کمتر است
- فرمول:

$$\text{Specificity} = \text{TN} / (\text{FP} + \text{TN})$$

- **Trade-off بین Sensitivity و Specificity**

- حساسیت بالا → شناسایی بیشتر هویت‌های درست (کمتر **false negative** ولی احتمال بیشتر اشتباه مثبت (**more false positives**))
- حساسیت پایین → اشتباه مثبت کمتر (کمتر **false positives** ولی احتمال رد هویت‌های درست بیشتر (**more false negatives**))
- **اثر روی سیستم احراز هویت:** باید تعادلی پیدا کرد بین قبول کردن افراد واقعی و رد نکردن افراد واقعی، تا امنیت و راحتی کاربر حفظ شود.

- **Accuracy** (کارایی کلی): درصد شناسایی درست هویت‌ها  

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN}$$
- **Prevalence** (شیوع): نسبت واقعی مثبت‌ها در کل نمونه‌ها  

$$\text{Prevalence} = \frac{TP+FN}{TP+FP+TN+FN}$$
- **Positive Predictive Value (PPV)**: احتمال اینکه تشخیص مثبت درست باشد  

$$\text{PPV} = \frac{TP}{TP+FP}$$
- **Negative Predictive Value (NPV)**: احتمال اینکه تشخیص منفی درست باشد  

$$\text{NPV} = \frac{TN}{TN+FN}$$

- **ROC Curve**: نمودار نرخ مثبت واقعی (True Positive Rate) در مقابل نرخ مثبت کاذب (False Positive Rate) برای هر مقدار آستانه.
- **دقت تست**: متناظر با مساحت زیر منحنی (AUC)
  - $AUC = 1$  → تست کامل و بی نقص
  - $AUC = 0.5$  → تست بی فایده

- **مشکل‌ها در احراز هویت بیومتریک**:
  - امکان جعل وجود دارد (مثلاً اثر انگشت می‌تواند با masterprints فریب داده شود).
  - ویژگی‌های بیومتریک ممکن است با گذر زمان تغییر کنند → دقت کمتر نسبت به احراز هویت مبتنی بر دانش.

- **توکن (Token)**: چیزی که در اختیار کاربر است برای احراز هویت.
  - مثال‌ها: کلید، کارت شناسایی، کارت اعتباری با نوار مغناطیسی یا چیپ، کارت دسترسی، گوشی هوشمند با تکنولوژی بی سیم فعال/غیرفعال.

- **توکن استاتیک**: مقدار ثابت دارد، راحت کپی یا جعل می‌شود.
- **توکن دینامیک**: مقدارش تغییر می‌کند، هر مقدار فقط برای یک بار دسترسی معتبر است و کسی با دانستن مقدار قبلی نمی‌تواند مقدار بعدی را حدس بزند.

- **RSA SecurID**: توکن هر ۶۰ ثانیه یک کد یکتا و حساس به زمان تولید می‌کند.
- کد با استفاده از **کلید مخفی مخصوص هر توکن و زمان جاری** ساخته می‌شود.

- (MFA) احراز هویت چند عاملی: (نیاز به ارائه دو یا چند عامل تأیید هویت دارد).
- (2FA) دو عاملی: (نمونه: وارد کردن رمز عبور (چیزی که می دانید) و دریافت کد یک بار مصرف از طریق پیامک یا ایمیل (چیزی که دارید)).

- **Single Sign-On (SSO):**
  - احراز هویت با یک SSO ID برای چند نرم افزار مرتبط اما مستقل.
  - کاربر یک بار لاگین می کند و به چند سیستم بدون ورود مجدد دسترسی پیدا می کند.
  - معمولاً در یک سازمان یا دامنه امنیتی استفاده می شود.
- **Federated Identity Management:**
  - گسترش SSO بین چند سازمان یا دامنه امنیتی.
  - کاربر یک بار احراز هویت می کند و به سرویس های مختلف سازمان ها دسترسی پیدا می کند.
- مثال پروتکل ها: SAML، OAuth، OpenID Connect، Kerberos.

اسلاید سوم:

- **مسئله امنیتی:**
  - **علیس** → **باب**: ارسال پیام
  - **ایو**: نفوذگر که می تواند:
    - پیام را **استراق سمع** کند → **نقض محرمانگی**
    - پیام را **تغییر** دهد → **نقض تمامیت**
    - پیام را **مسدود** کند → **نقض دسترس پذیری**
- **راه حل:**
  - **رمزنگاری** برای حفظ امنیت داده ها در محیط ناامن.

- **Encryption:** فرآیند رمزگذاری پیام تا معنی آن واضح نباشد.
- **Decryption:** تبدیل پیام رمز شده به شکل اصلی و قابل فهم.
- **Cryptography:** طراحی، تحلیل و استفاده از الگوریتم ها برای رمزگذاری و رمزگشایی.
- **Plaintext:** شکل اصلی و قابل فهم پیام.
- **Ciphertext:** شکل رمز شده پیام.



- الگوریتم‌های رمزگذاری و رمزگشایی معمولاً از یک کلید استفاده می‌کنند.
- متن رمز شده (Ciphertext) به متن اصلی، الگوریتم و کلید وابسته است.
- بهتر است الگوریتم‌های رمزگذاری شناخته شده و امن استفاده شوند، اما کلیدها متفاوت باشند تا امنیت حفظ شود.
- معادلات:

$$C = E(K, P) \rightarrow \text{رمزگذاری: متن اصلی} + \text{کلید} \rightarrow \text{متن رمز شده}$$

$$P = D(K, C) \rightarrow \text{رمزگشایی: متن رمز شده} + \text{کلید} \rightarrow \text{متن اصلی}$$

- امنیت رمزگذاری به محرمانه بودن کلید بستگی دارد، نه الگوریتم.
- الگوریتم باید قوی باشد تا حتی اگر حریف الگوریتم را بداند، نتواند متن رمز شده را رمزگشایی یا کلید را پیدا کند.
- امکان پیاده‌سازی الگوریتم‌های رمزگذاری با هزینه پایین روی چیپ وجود دارد.
- رمزگذاری متقارن: فرستنده و گیرنده از یک کلید یکسان استفاده می‌کنند (کلید مخفی).
- رمزگذاری نامتقارن: فرستنده و گیرنده هر کدام کلید متفاوتی دارند (کلید عمومی و خصوصی).

- همه الگوریتم‌های رمزگذاری روی دو اصل پایه‌اند:
  - جابجایی (Transposition): عناصر پیام جابه‌جا می‌شوند.
  - جانشینی (Substitution): هر عنصر (مثل هر بایت) به عنصر دیگری تبدیل می‌شود.

- Stream cipher: داده‌ها به صورت پیوسته و یکی یکی پردازش می‌شوند.
  - مناسب برای ارسال فوری داده‌ها
  - خطای کمتر در انتشار
- Block cipher: داده‌ها به بلوک‌های مشخصی تقسیم و پردازش می‌شوند.
  - امنیت بالاتر (diffusion) بیشتر
  - اندازه بلوک‌ها معمولاً 64، 128، 256 بیت یا بیشتر

- **Cryptanalysis:** تلاش برای کشف متن اصلی یا کلید رمز.

#### انواع حمله:

- **Ciphertext-only:** الگوریتم رمز + متن رمز؛ متن اصلی ناشناخته
- **Known-plaintext:** الگوریتم رمز + متن رمز + یک یا چند جفت متن اصلی-رمز
- **Chosen-plaintext:** الگوریتم رمز + متن رمز + متن اصلی انتخاب شده توسط حمله کننده و متن رمز مربوطه
- **Chosen-ciphertext / Chosen-text:** مشابه بالا با امکان انتخاب متن رمز برای تحلیل

- **رمزگذاری محاسباتی امن:**

رمزگذاری زمانی امن است که:

- هزینه شکستن رمز بیشتر از ارزش اطلاعات رمزگذاری شده باشد
- یا زمان لازم برای شکستن رمز بیشتر از عمر مفید اطلاعات باشد

- **Work factor** بزرگ = بازدارنده برای بیشتر حملات

- **تخمین تلاش برای کرایپتانالیز:**

- فرض: الگوریتم ضعیف ریاضی ندارد → فقط حمله brute-force ممکن است
- **حمله: brute-force** امتحان همه کلیدهای ممکن تا رسیدن به متن قابل فهم
- به طور متوسط، نصف کلیدها باید امتحان شوند
- بدون متن معلوم، تحلیلگر باید بتواند متن به دست آمده را به عنوان plaintext شناسایی کند

- **DES (Data Encryption Standard):**

- توسعه یافته در دهه ۱۹۷۰، استاندارد NIST
- الگوریتم رمزنگاری متقارن
- **Block cipher** با اندازه بلوک ۶۴ بیت
- طول کلید: ۵۶ بیت → امروز ناامن
- **Triple DES:** اجرای سه باره DES با سه کلید ۵۶ بیتی → قدرت برابر با تقریباً کلید ۱۱۲ بیتی

- DES کند است و بلوک ۶۴ بیتی از نظر کارایی و امنیت کافی نیست
- سال ۱۹۹۷، NIST فراخوان داد برای AES به عنوان جایگزین:
  - قدرت امنیتی برابر یا بهتر از DES3
  - کارایی بالاتر
  - معیارهای ارزیابی: امنیت، سرعت محاسباتی، مصرف حافظه، مناسب برای سخت افزار و نرم افزار، انعطاف پذیری

### • AES (Rijndael)

- توسعه یافته توسط Vincent Rijmen و Joan Daemen ، استاندارد ۲۰۰۱
- الگوریتم رمزنگاری متقارن
- بلوک سائز: ۱۲۸ بیت
- اندازه کلید: ۱۲۸، ۱۹۲، ۲۵۶ بیت

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at $10^9$ Decryptions/s	Time Required at $10^{13}$ Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55} \text{ ns} = 1.125 \text{ years}$	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127} \text{ ns} = 5.3 \times 10^{21} \text{ years}$	$5.3 \times 10^{17} \text{ years}$
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167} \text{ ns} = 5.8 \times 10^{33} \text{ years}$	$5.8 \times 10^{29} \text{ years}$
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191} \text{ ns} = 9.8 \times 10^{40} \text{ years}$	$9.8 \times 10^{36} \text{ years}$
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255} \text{ ns} = 1.8 \times 10^{60} \text{ years}$	$1.8 \times 10^{56} \text{ years}$

### • Message Authentication

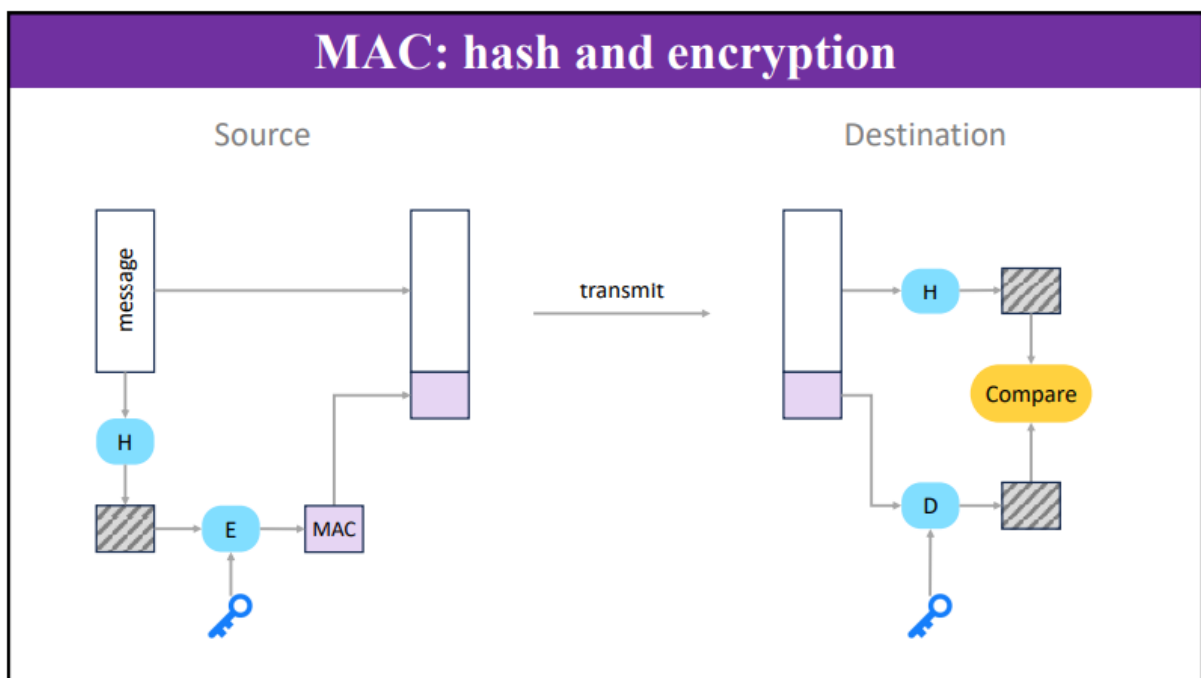
- محافظت در برابر حملات فعال
- تضمین می کند پیام:
  - در حین انتقال تغییر نکرده باشد (یکپارچگی داده)
  - از منبع معتبر ارسال شده باشد (اصالت)
  - تازه باشد و تکراری از پیام قدیمی نباشد (تازگی)

- راه حل برای احراز پیام:

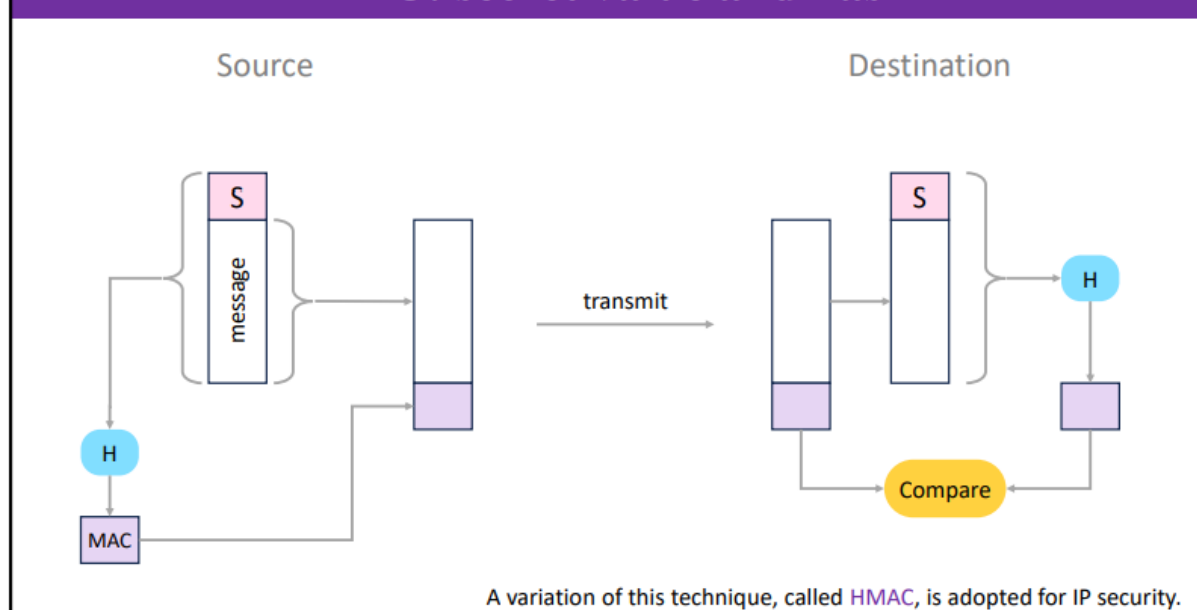
- از یک کلید مخفی استفاده می‌کنیم تا یک بلوک کوچک داده به نام **MAC** (Message Authentication Code) تولید شود و به پیام اضافه شود
- فرمول  $MAC_m = F(m, k)$  :

• روش‌های تولید: **MAC**

- استفاده از الگوریتم رمزنگاری متقارن و گرفتن آخرین بیت‌های رمزنگاری پیام → ناکارآمد برای پیام‌های بزرگ
- استفاده از تابع هش یک‌طرفه → محاسبات کمتر و سریع‌تر



## MAC: secret value and hash



ویژگی‌های ضروری برای تابع هش در احراز صحت پیام:

- سریع و آسان برای محاسبه
- یک‌طرفه: یافتن  $x$  از روی  $H(x) = h$  غیرممکن است
- مقاومت در برابر برخورد ضعیف: یافتن  $x \neq y$  با  $H(x) = H(y)$  سخت است
- مقاومت در برابر برخورد قوی: یافتن هر جفت  $(x, y)$  با  $H(x) = H(y)$  غیرممکن است

حملات روی توابع هش:

- تحلیل رمزنگاری: استفاده از ضعف‌های منطقی الگوریتم
- حمله جستجوی فراگیر (brute-force)

قدرت توابع هش در برابر brute-force بسته به طول خروجی  $n$  است:

- مقاومت در برابر پیش‌تصویر  $2^n$ : (preimage resistant)
- مقاومت در برابر پیش‌تصویر دوم  $2^n$ : (second preimage resistant)
- مقاومت در برابر برخورد -  $2^{\{n/2\}}$ : (collision resistant) حمله تولد (Birthday attack)

توابع هش امن:

- 128 (1991) MD5 بیت، امروزه ناامن

- 160 (1995): SHA-1 بیت
- 256 → SHA-256 (2001): SHA-2 بیت، 512 → SHA-512 بیت

### HMAL:

- توابع هش معمولی مثل SHA-1 بدون کلید مخفی برای MAC مناسب نیستند.
- $HMAL =$  ترکیب تابع هش رمزنگاری با یک کلید مخفی
- می‌توان از SHA-2 یا SHA-3 در محاسبه HMAL استفاده کرد.

### محدودیت‌های رمزنگاری متقارن:

- توزیع کلید: چگونه دو طرف کلید مخفی را امن به اشتراک بگذارند؟
- مقیاس‌پذیری: در شبکه با  $n$  کاربر، هر جفت نیاز به یک کلید منحصر به فرد دارد →  $n(n-1)/2$  کلید برای مدیریت.
- احراز هویت: رمزنگاری متقارن به تنهایی راهی برای تأیید هویت فرستنده ندارد (برای ارتباط با طرف ناشناخته).

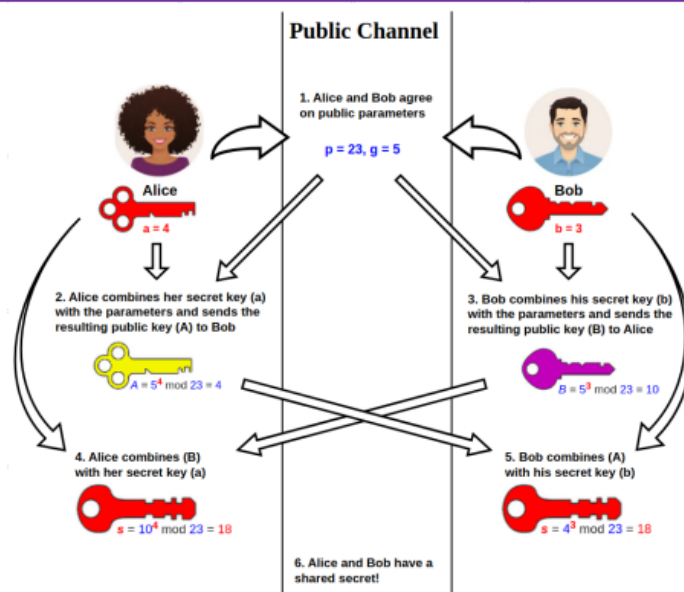
### رمزنگاری نامتقارن (کلید عمومی):

- ابداع‌شده توسط Diffie و Hellman در 1976.
- هر کاربر دو کلید دارد: یک کلید عمومی و یک کلید خصوصی.
- کلید عمومی را می‌توان آزادانه منتشر کرد، اما کلید خصوصی باید مخفی بماند.
- از روی یک کلید، یافتن کلید دیگر عملاً غیرممکن است.
- امنیت سیستم به مخفی ماندن کلید خصوصی وابسته است.
- امنیت بر پایه مسائل ریاضی دشوار و محاسباتی سخت است.

### الگوریتم تبادل کلید:

- هر طرف یک جفت کلید عمومی/خصوصی تولید و کلید عمومی را منتشر می‌کند.
- پس از دریافت نسخه معتبر کلید عمومی طرف مقابل، هر طرف می‌تواند یک راز مشترک به‌صورت آفلاین محاسبه کند.
- این راز مشترک می‌تواند به‌عنوان کلید یک رمزنگاری متقارن استفاده شود.

## Diffie-Hellman algorithm



### RSA:

- در 1977 Rivest, Shamir و Adleman ساخته شده توسط.
- سیستم رمزنگاری کلید عمومی شامل تولید کلید، رمزگذاری و رمزگشایی.
- امنیت بر اساس دشواری عملی فاکتورگیری حاصل ضرب دو عدد اول بزرگ است.
- رمزگذاری:  $C = E(K_{Public}, P)$
- رمزگشایی:  $P = D(K_{Private}, C)$

### رمزنگاری کلید عمومی با: RSA

- هر کاربر یک جفت کلید تولید می کند: عمومی + خصوصی.
- کلید عمومی در یک رجیستر عمومی قرار می گیرد یا منتشر می شود.
- برای ارسال پیام خصوصی به Alice، Bob پیام را با کلید عمومی Alice رمزگذاری می کند.
- Alice پیام را با کلید خصوصی خودش رمزگشایی می کند.
- فقط Alice می تواند پیام را بخواند؛ دیگران قادر به رمزگشایی نیستند.

### ویژگی های سیستم کلید عمومی و: RSA

- هر کاربر فقط به دو کلید نیاز دارد.
- RSA نسبتاً کند است.
- طول کلیدها زیاد است: معمولاً 2048، 3072 و 4096 بیت.
- رمزگذاری با توان برداری انجام می شود (هر بلوک متن اصلی به توان کلید می رسد).

- معمولاً RSA برای رمزگذاری مستقیم داده‌ها استفاده نمی‌شود؛ بیشتر برای انتقال کلیدهای مشترک در رمزنگاری متقارن به کار می‌رود.

### ویژگی‌های کلید عمومی در: RSA

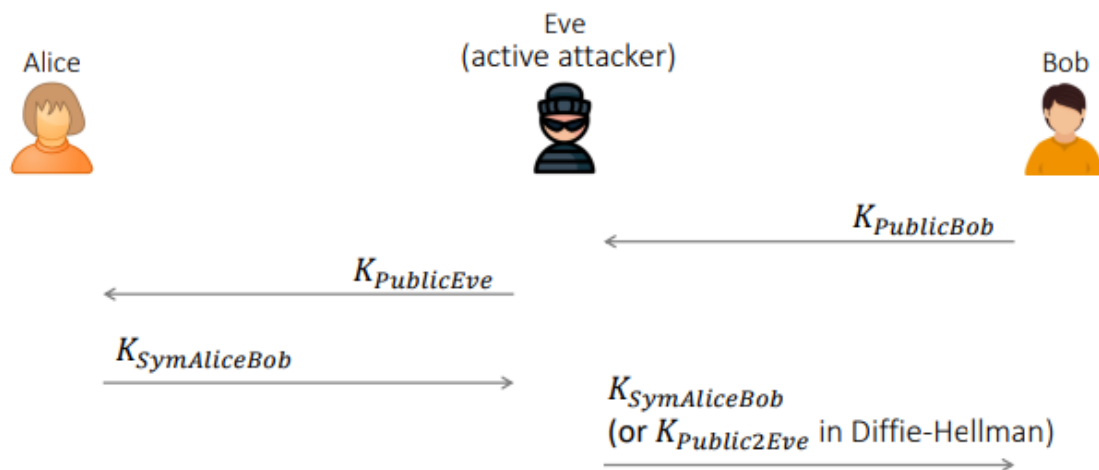
- کلید عمومی و خصوصی را می‌توان به ترتیب دلخواه اعمال کرد.
- کاربردها:
  - رمزنگاری با کلید عمومی
  - امضای دیجیتال

$$P = D(K_{\text{Private}}, E(K_{\text{Public}}, P)) \rightarrow \text{Application: public-key encryption}$$

$$P = D(K_{\text{Public}}, E(K_{\text{Private}}, P)) \rightarrow \text{Application: digital signature}$$

### حمله مرد میانی: (Man-in-the-Middle)

- حمله‌ای که در آن مهاجم (Eve) در میانه ارتباط بین Alice و Bob قرار می‌گیرد.
- کلیدهای عمومی تبادل شده را دستکاری می‌کند و خودش را جای یکی از طرفین جا می‌زند.
- پروتکل‌های تبادل کلید نامتقارن در صورت عدم احراز هویت شرکت کنندگان آسیب‌پذیرند.



امضا: (Signature)



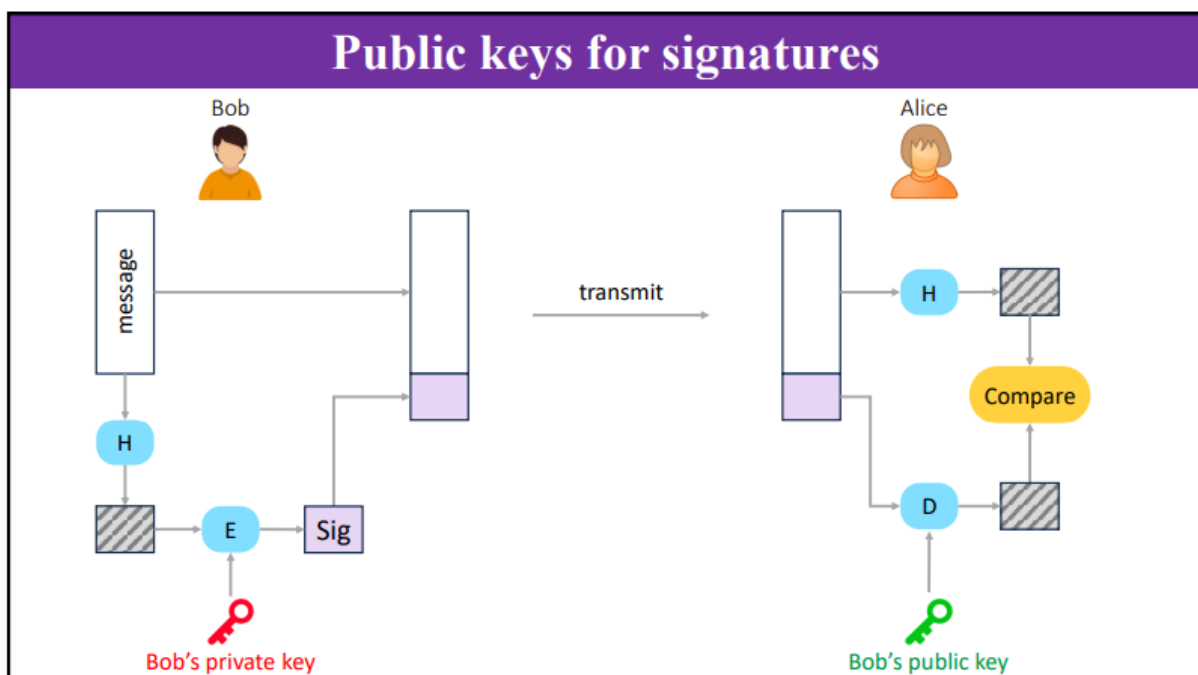
- علامتی که فقط فرستنده می‌تواند بسازد، ولی دیگران به راحتی آن را به فرستنده نسبت می‌دهند.
- کاربرد:
  - احراز هویت (Authenticity)
  - عدم انکار (Non-repudiation)

### امضای دیجیتال:

- یک شیء باینری متصل به پیام که تأیید می‌کند فرستنده با پیام موافق است.
- دو شرط اصلی:
  - غیرقابل جعل (Unforgeable) هیچ کس جز فرستنده نمی‌تواند امضای پیام را تولید کند.
  - اصالت (Authentic) گیرنده می‌تواند مطمئن شود امضا واقعاً از فرستنده است.

### ویژگی‌های اضافی امضای دیجیتال:

- غیرقابل تغییر: پس از ارسال، پیام توسط فرستنده، گیرنده یا فرد ثالث قابل تغییر نیست.
- غیرقابل استفاده مجدد: ارائه مجدد یک پیام قبلی به سرعت توسط گیرنده شناسایی می‌شود.

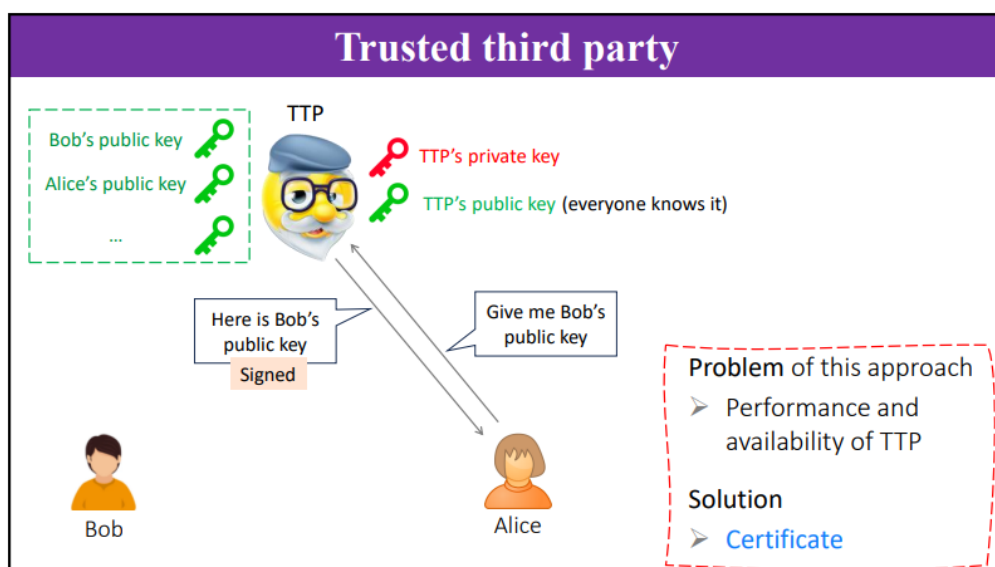


کلیدهای عمومی برای امضا:

- فقط کلید خصوصی باب می‌تواند امضا ایجاد کند که با کلید عمومی او تأیید شود.
- بدون دسترسی به کلید خصوصی باب، امکان تغییر پیام وجود ندارد.
- برای حفظ محرمانگی، باب می‌تواند کل پیام را با کلید عمومی آلیس رمزنگاری کند.

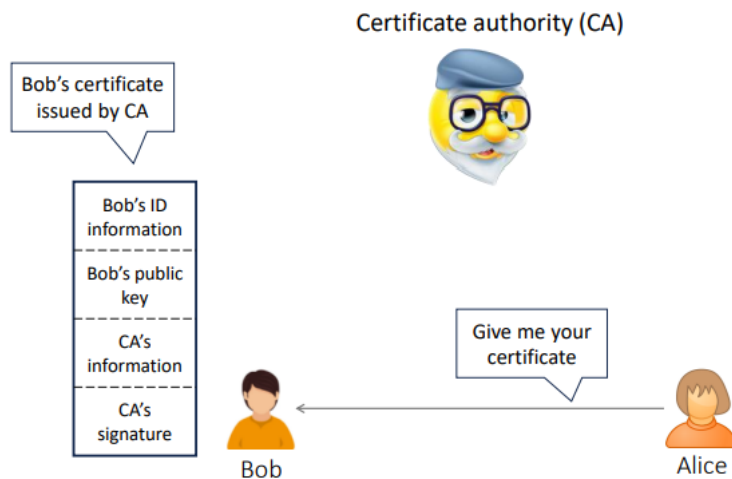
#### مسئله اعتماد:

- مشکل: وقتی گفته می‌شود «این کلید عمومی به باب تعلق دارد»، چگونه به آن اعتماد کنیم؟
- راهکار روزمره: برای کسی که هرگز ملاقات نکرده‌ایم، به یک طرف سوم مورد اعتماد متکی می‌شویم.

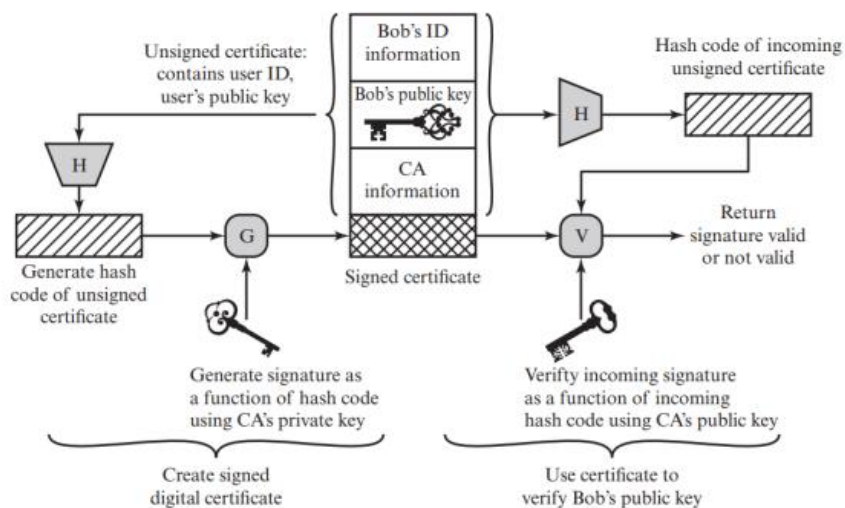


#### گواهی دیجیتال: (Certificate)

- شامل کلید عمومی + شناسه کاربر، که توسط یک طرف سوم مورد اعتماد امضا شده است.
- طرف سوم: مرجع صدور گواهی (CA) که برای کاربران معتبر است.
- کاربر کلید عمومی خود را به CA ارائه و گواهی دریافت می‌کند.
- گواهی منتشر می‌شود و دیگران می‌توانند با بررسی امضای CA، صحت آن را تأیید کنند.



## Certificate



### مرجع صدور گواهی ریشه: (Root CA)

- چند شرکت معتبر به عنوان **Root CA** شناخته می‌شوند:
  - Staat der ،Baltimore Technologies ،VeriSign ،SecureNet
  - Certiposte ،Entrust ،Deutsche Telecom ،Nederlanden
- مرورگرها و سیستم‌عامل‌ها به گواهی‌های صادرشده توسط این CA های معتبر اعتماد می‌کنند.

### ( PKI زیرساخت کلید عمومی: )

- مجموعه‌ای از ابزارها، نقش‌ها، قوانین و روش‌ها برای ایجاد، مدیریت، توزیع، استفاده، ذخیره و لغو **گواهی‌های دیجیتال** و مدیریت رمزنگاری کلید عمومی.
- از **گواهی‌ها** برای تبادل امن کلیدها با اعتماد استفاده می‌کند.
- کاربرد اصلی: جریان داده در اینترنت، شامل پیام‌رسانی و تراکنش‌ها بین کاربران و وب‌سایت‌ها.

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Diffie–Hellman	No	No	Yes
DSS	No	Yes	No
Elliptic curve	Yes	Yes	Yes

اسلاید چهارم:

### مشکلات امنیتی در برنامه‌ها:

- ناشی از خطاهای ناخواسته یا عمدی در برنامه‌نویسی.
- حمله‌کننده‌های مخرب می‌توانند از خطاهای غیرعمدی سوءاستفاده کنند.
- خطاهای ساده برنامه‌نویسی می‌توانند منجر به مشکلات بزرگ در اجرا شوند.

### عیوب برنامه و پیامدهای امنیتی:

- می‌توانند باعث نقض یکپارچگی و تولید خروجی یا عمل مضر شوند.
- می‌توانند فرصت سوءاستفاده توسط مهاجم ایجاد کنند.

### نمونه‌های عیب برنامه‌ای با پیامد امنیتی:

- Buffer overflow
- Incomplete mediation
- Off-by-one error
- Unsafe utility program
- Time-of-check to time-of-use (TOCTOU)
- Undocumented access point
- Integer overflow
- Unterminated null-terminated string

## Buffer Overflow: و Buffer

- **Buffer:** فضای حافظه‌ای برای نگهداری داده‌ها.
- **Buffer overflow:** وقتی برنامه داده‌ای فراتر از حافظه تخصیص داده شده به بافر می‌نویسد و باعث بازنویسی حافظه‌های مجاور می‌شود.

## اثرات: Buffer Overflow

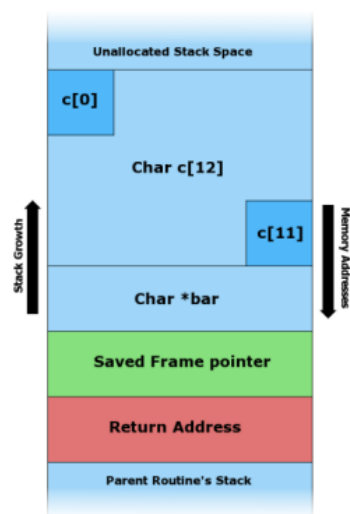
- آسیب یا سوءاستفاده می‌تواند بسیار بزرگ‌تر از خطای اولیه باشد.
- مهاجم می‌تواند از آن برای اهداف مخرب استفاده کند.
- داده‌های طراحی شده برای ایجاد **Buffer Overflow** می‌توانند:
  - به مناطق حافظه‌ای که شامل کد اجرایی هستند بنویسند و آن را با کد مخرب جایگزین کنند.
  - داده‌های مرتبط با وضعیت برنامه را بازنویسی کنند و باعث رفتار غیرمنتظره برنامه شوند.

## Buffer overflow attack example: stack overflow

```
#include <string.h>

void foo(char *bar)
{
    char c[12];
    strcpy(c, bar); // no bounds checking
}

int main(int argc, char **argv)
{
    foo(argv[1]);
    return 0;
}
```



Stack

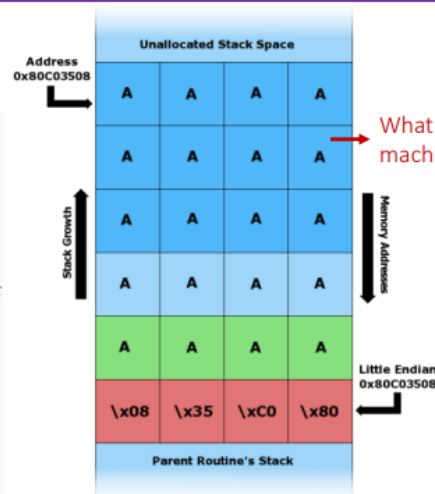
## Buffer overflow attack example: stack overflow

```
#include <string.h>

void foo(char *bar)
{
    char c[12];

    strcpy(c, bar); // no bounds check
}

int main(int argc, char **argv)
{
    foo(argv[1]);
    return 0;
}
```



What would happen if these were machine-executable codes?

Command line argument:  
"AAAAAAAAAAAAAAAAAAAA  
A\x08\x35\xC0\x80"

Solution?

### Heap Overrun / Stack Smashing:

- Stack و Heap به سمت هم رشد می کنند و ممکن است برخورد کنند.
- هدف مهاجم: بازنویسی stack به صورت هدفمند.
  - بازنویسی Program Counter در → stack هنگام خروج از تابع، کنترل به آدرس تغییر یافته می رود.
  - بازنویسی Program Counter و داده ها در Program Counter → stack به stack اشاره می کند و داده های بازنویسی شده اجرا می شوند.

### Privilege Escalation:

- برنامه ها توسط OS اجرا می شوند که ممکن است دسترسی بالاتری نسبت به برنامه عادی داشته باشد.
- مهاجم با بازنویسی چند دستور بعد از بازگشت از تابع خود، کنترل را از OS بازی گیرد و ممکن است دسترسی ها را بالا ببرد.
- اگر buffer به فضای کد سیستم overflow کند → مهاجم داده هایی وارد می کند که معادل کد ماشین برای دستورات مورد نظر هستند.

### Buffer Overflow via Parameter Passing:

- رخ می دهد وقتی مقادیر پارامتر به یک تابع یا سرویس ارسال می شوند.
- مثال: پارامتر URL برای وب سرور → اگر شماره تلفن خیلی طولانی (مثلاً 500 یا 1000 رقم) باشد، ممکن است باعث overflow شود.

## Unchecked Buffer Bounds:

- در برخی زبان‌ها اندازه بافر لازم نیست از قبل تعریف شود → تشخیص خطای out-of-bounds ممکن نیست.
- بررسی هر اندیس با مقدار ماکزیمم نیازمند زمان و حافظه است → منابع صرف مشکلی می‌شود که به ندرت رخ می‌دهد.
- حتی اگر کامپایلر با دقت بررسی کند، استفاده از pointers می‌تواند همان مشکل را ایجاد کند.
- بنابراین، برخی کامپایلرها کد بررسی تجاوز از حد بافر را تولید نمی‌کنند.

## راهکارهای مقابله با بافر اورفلو:

- قبل از نوشتن، طول داده‌ها را بررسی کن.
- مطمئن شو اندیس‌های آرایه در محدوده هستند.
- ورودی‌ها را کنترل کن و فقط به اندازه‌ای که می‌توان مدیریت کرد بپذیر.
- زبان‌هایی مثل Java، NET، Perl و Python به طور خودکار بررسی bounds انجام می‌دهند.
- زبان‌هایی مثل C، C++ و اسمبلی دسترسی نامحدود برنامه را ممکن می‌کنند.

## Incomplete mediation:

- بررسی اینکه موضوع (subject) اجازه انجام عملیات روی شیء (object) را دارد، mediation نامیده می‌شود.
- حمله‌کنندگان با سوءاستفاده از mediation ناقص مشکلات امنیتی ایجاد می‌کنند.

- ارسال مقادیر نامعتبر (مثلاً تاریخ‌های اشتباه یا رشته‌های غیرمنتظره) می‌تواند:
  - باعث شکست کامل سیستم شود، یا
  - برنامه ادامه دهد اما نتیجه بسیار اشتباه تولید کند.

نکات مهم:

- برنامه‌نویسان معمولاً ورودی‌های نامعتبر را جدی نمی‌گیرند.
- فقط به خاطر بی‌معنی یا اشتباه بودن داده‌ها، کاربران از وارد کردن آن‌ها باز نمی‌مانند.
- برنامه‌ها باید تمام ورودی‌ها را قبل از استفاده بررسی و اعتبارسنجی کنند تا مطمئن شوند منطقی هستند.

### نکات مهم:

- بررسی و اعتبارسنجی داده‌ها می‌تواند در سمت کاربر (**client**) انجام شود.
- برنامه می‌تواند خطاها را تشخیص دهد و ورودی‌های نامعتبر را حذف کند.
- برای جلوگیری از داده‌های نامعتبر، می‌توان انتخاب‌ها را محدود کرد، مثل استفاده از drop-down list با ۱۲ ماه معتبر.

### نکات مهم:

- برنامه هنوز آسیب‌پذیر است چون کاربر می‌تواند URL را تغییر دهد و پارامترها را دستکاری کند.
- داده‌ها به‌طور کامل کنترل نشده‌اند. (**incomplete mediation**)
- راه‌حل **Complete mediation**: یعنی همه داده‌ها باید قبل از استفاده بررسی و تایید شوند.

- **TOCTOU flaw** خطا در mediation وقتی بین بررسی و استفاده از داده، وقفه وجود دارد.
- حمله از تأخیر بین **check** و **use** بهره می‌برد.
- داده‌ها باید بین زمان بررسی و زمان استفاده محافظت شوند.

❓ درخواست دسترسی به فایل به صورت **data structure** ارائه می‌شود (نام فایل + نوع دسترسی).

❓ پس از تأیید، دسترسی در **queue** قرار می‌گیرد.

❓ بین زمان بررسی (**check**) و استفاده (**use**) تغییر رخ می‌دهد و نتیجه بررسی بی‌اعتبار می‌شود.



## Example

- In Unix, the following C code, when used in a `setuid` program, has a TOCTOU bug

```
1  if (access("file", W_OK) != 0) {  
2      exit(1);  
3  }  
4  
5  fd = open("file", O_WRONLY);  
6  write(fd, buffer, sizeof(buffer));
```

- After the access check, before the open, the attacker replaces file with a symlink to the Unix password file `/etc/passwd`

- هرگاه تاخیر زمانی یا از دست دادن کنترل وجود دارد، باید مراقب باشیم که نتیجه بررسی دستکاری نشود.
- راهکارها:
  - داده‌ها را از فضای کاربر کپی کرده و بررسی روی نسخه کپی انجام شود.
  - داده‌های درخواست را seal کنیم تا تغییرات شناسایی شود.
- در توسعه برنامه، برنامه‌نویس‌ها گاهی نیاز به دسترسی به داخلی ماژول دارند.
- برای این کار، نقاط ورود یا حالت‌های اجرا بدون مستندسازی ایجاد می‌کنند.
- این نقاط ورود بدون مستندسازی **backdoor** یا **trapdoor** نامیده می‌شوند.
- این نقاط می‌توانند کنترل را به هر نقطه‌ای با هر سطح دسترسی منتقل کنند.
- گاهی حذف این نقاط هنگام عرضه فراموش می‌شود یا عمداً برای نگهداری برنامه باقی گذاشته می‌شوند.
- (Malware نرم‌افزار مخرب): نرم‌افزاری که عمداً برای اختلال در کامپیوتر، افشای اطلاعات خصوصی، دسترسی غیرمجاز یا محروم کردن دسترسی به اطلاعات طراحی شده است.
- شایع‌ترین انواع Virus، Worm، Trojan horse.
- Virus: نوعی malware که با اجرا شدن، خودش را تکثیر می‌کند و با تغییر برنامه‌های غیرمخرب، کد خودش را در آن‌ها وارد می‌کند.
- نیاز به برنامه میزبان دارد و برنامه آلوده می‌تواند دیگر برنامه‌ها را هم آلوده کند.

- Worm: نوعی malware مستقل که خودش را تکثیر می کند تا به کامپیوترهای دیگر منتقل شود.
- معمولاً از شبکه برای انتشار استفاده می کند و از ضعف های امنیتی سیستم هدف سوءاستفاده می کند.

- Trojan: برنامه ای که ظاهراً مفید است اما اثر مخرب پنهانی دارد.
- روی سطح، برنامه کاربردی است اما شامل ویژگی های مخفی و مخرب می شود.
- لزوماً سعی در تکثیر خود ندارد.

- راه های انتقال و انتشار بدافزار:
  - برنامه نصب و راه انداز
  - فایل ضمیمه شده به پیام
  - ویروس های سندی
  - اجرای خودکار (Autorun)

- ویروس های الحاقی: (Appended virus)
  - طراحی و پیاده سازی آسان
  - نویسنده ویروس نیازی به شناخت برنامه میزبان ندارد
  - برنامه میزبان معمولاً فقط به عنوان حامل ویروس عمل می کند

- ویروس های یکپارچه: (Integrated virus)
  - نویسنده ویروس باید ساختار دقیق برنامه میزبان را بداند
  - می داند کدام بخش ویروس را کجا وارد کند

- چالش برنامه های ضد ویروس:
  - ابزارها معمولاً گذشته نگر هستند و به دنبال الگوهای ویروس های شناخته شده می گردند

- ویروس‌ها را با اسکن برنامه‌ها و جستجوی امضاها ویروس شناسایی می‌کنند
- با ظهور ویروس‌های جدید، الگوها باید مرتب به‌روز شوند
- ویروس‌های تازه ممکن است هنوز در فایل‌های الگو نباشند و شناسایی نشوند

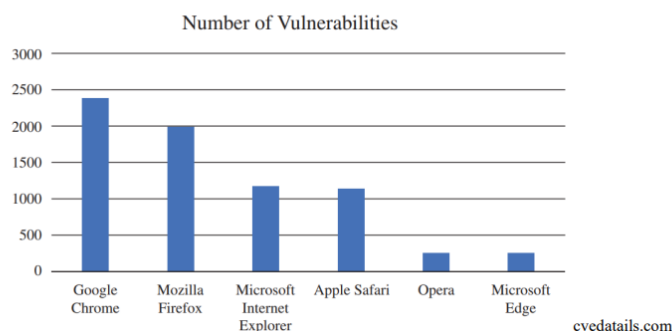
- ویروس‌ها و کرم‌های چندشکل (Polymorphic) در هر بار آلوده‌سازی کد خود را تغییر می‌دهند
- برنامه‌های ضدویروس از تحلیل هورستیک استفاده می‌کنند تا ویروس‌های ناشناخته و واریانت‌های جدید ویروس‌های موجود را شناسایی کنند
- فقط از نرم‌افزارهای تجاری و معتبر استفاده کن
- نرم‌افزار جدید را ابتدا روی یک سیستم ایزوله تست کن
- فایل‌های ضمیمه را فقط وقتی مطمئنی ایمن هستند باز کن
- همه وب‌سایت‌ها ممکن است خطرناک باشند
- یک تصویر قابل بازیابی از سیستم بساز و آن را ایمن نگه دار

#### اسلاید پنجم)

- یک صفحه وب می‌تواند شامل لینک‌هایی به صفحات دیگر باشد: آدرس ظاهری قابل اعتماد نیست
- مرورگر یا صفحات وب می‌توانند آلوده به عملکردهای مخرب شوند
- مرورگرهای محبوب از افزونه‌ها (add-ons) پشتیبانی می‌کنند که ممکن است شامل کد مخرب باشند
- مرورگر می‌تواند به داده‌های کاربر دسترسی پیدا کند (با محدودیت‌های کنترل دسترسی)
- برخی انتقال‌های داده به صورت خودکار و بدون اطلاع یا اجازه کاربر انجام می‌شوند

## Browsers' security concerns

The cumulative number of vulnerabilities discovered to 2022 in the major browsers



### Man in the browser:

- کد مخرب که مرورگر را آلوده می کند
- کد وارد شده به مرورگر می تواند اطلاعات وارد شده توسط کاربر را بخواند، کپی کند و پخش کند
- تهدید اصلی: رهگیری و استفاده دوباره از اطلاعات ورود برای دسترسی به حساب های مالی و داده های حساس
- مثال SilentBanker: 2008

### Silent banker Trojan:

- کد به مرورگر قربانی به صورت افزونه یا Browser Helper Object متصل می شود
- به عنوان Helper Object، تماس های داخلی مرورگر را برای دریافت داده از کیبورد، ارسال داده به URL، ایجاد یا وارد کردن کلید رمزنگاری، خواندن فایل یا اتصال به سایت و دانلود داده ها رهگیری می کرد
- هنگام رفتن کاربر به سایت بانکی، ضربه های کلید کاربر را از طریق تروجان هدایت و جزئیات مشتری را به کامپیوترهای راه دور ارسال می کرد
- تراکنش های بانکی معمولاً با یک نشست رمزگذاری شده محافظت می شوند، با استفاده از پروتکل SSL یا HTTPS و با نماد قفل روی مرورگر مشخص می شوند
- رمزنگاری تنها می تواند داده هایی را محافظت کند که تحت کنترل آن باشد
- رمزگذاری SSL در مرورگر اعمال می شود؛ داده ها قبل از رمزگذاری آسیب پذیر هستند
- SilentBanker اثر اقدامات مشتری را تغییر می داد
- حملات Man-in-the-browser می توانند مخرب باشند چون با کاربر معتبر و احراز هویت شده تعامل می کنند و تروجان بین کاربر و وبسایت بانک قرار می گیرد، به طوری که محتوای بانک همچنان واقعی به نظر می رسد

- کی لاگر (Keystroke logger) سخت افزاری یا نرم افزاری است که ضربه های کلید وارد شده توسط کاربر را ضبط می کند
- کی لاگر ممکن است این ضربه ها را برای استفاده بعدی توسط حمله کننده ذخیره کند یا فوراً از طریق شبکه به او ارسال کند

#### Page in the middle:

- کاربر به صفحه ای دیگر هدایت می شود
- مثال: هنگام کلیک روی [login] ، کاربر به صفحه حمله کننده هدایت می شود و حمله کننده می تواند اطلاعات ورود کاربر را ضبط کند

- اینترنت بات نرم افزاری است که وظایف خودکار (اسکرپت ها) را در اینترنت اجرا می کند
- مثال: کمک به اسپمرها برای ثبت نام خودکار در حساب های ایمیل رایگان
- CAPTCHA یک معما است که تنها انسان می تواند آن را حل کند تا سرور بتواند بین انسان و برنامه خودکار تمایز قائل شود
- CAPTCHA مخفف Completely Automated Public Turing test to tell Computers and Humans Apart است

- اولین نوع CAPTCHA رشته ای از حروف و اعداد در شکل کج روی زمینه دانه دار بود
- این نوع با تکنیک های OCR مبتنی بر هوش مصنوعی به راحتی قابل شکستن است
- افزایش تحریف حروف چالش، تشخیص آن برای انسان واقعی دشوارتر می کند، به ویژه برای افراد با مشکلات بینایی
- نوع بعدی CAPTCHA شامل حل مسائل ساده کلامی بود، مثال: «سه بعلاوه دو چقدر می شود؟»

- Google reCAPTCHA به کاربران تصاویری نشان می دهد و از آن ها می خواهد همه تصاویری که ویژگی مشخصی دارند (مانند خودرو، چراغ راهنمایی، خط عابر پیاده، حیوانات یا پل) را علامت بزنند
- reCAPTCHA v2 با تحلیل رفتاری تعامل مرورگر، پیش بینی می کند که کاربر انسان است یا بات

- این نسخه از تکنیکی اختصاصی استفاده می کند که ویژگی های تعامل مانند مدت زمان وارد کردن داده یا محل توقف کاربر هنگام تایپ را بررسی می کند
- در reCAPTCHA v3 ، تأیید در پس زمینه انجام می شود و اگر کاربر کم ریسک باشد، هیچ چالشی نمایش داده نمی شود

حملاتی که کاربر رو هدف قرار میدن:

#### Website defacement:

- تغییر محتوای یک وبسایت زمانی رخ می دهد که مهاجم محتوای سایت قانونی را جایگزین یا تغییر دهد
- اهداف تغییر وبسایت متنوع است: شرمند کردن قربانیان، جلب توجه یا احترام، بیان نظرات سیاسی یا ایدئولوژیک
- مهاجم می تواند با استفاده از تغییر وبسایت، لینک ها را به مکان مخرب هدایت کند، مثال:
  - نمایش فرم ورود جعلی و دریافت نام کاربری و رمز عبور قربانی
  - جایگزینی محتوای قانونی (مثلاً لینک برنامه) با محتوای مخرب

- بررسی یکپارچگی با استفاده از Integrity checksum ، مثال Tripwire :
- کد یا داده های امضا شده:

Integrity checksum ➢ تضمینی به کاربر نمی دهد  
 ➢ امضای دیجیتال، مهر الکترونیکی است که صحت فایل یا داده را تضمین می کند

#### Fake website:

- مهاجم می تواند تمام تصاویر و منابع ظاهری یک وبسایت را دریافت کرده و نسخه جعلی با URL دیگر بسازد

#### Web bug:

- وب بیکن (Web beacon) تکنیکی است برای بررسی نامحسوس دسترسی کاربر به محتوا در صفحات وب و ایمیل
- معمولاً توسط تبلیغ کنندگان ثالث برای پایش فعالیت کاربران در وبسایت استفاده می شود
- وب بیکن معمولاً یک تصویر بسیار کوچک، به اندازه ۱ در ۱ پیکسل است

- نقاط کوچک عملیاتی به نام وب باگ‌ها می‌توانند الگوهای حرکت در صفحات را به نقاط جمع‌آوری مرکزی گزارش کنند و حریم خصوصی را به خطر بیندازند

#### Clickjacking:

- مهاجم کاربر را فریب می‌دهد تا بدون اطلاع یا قصد خود روی چیزی در وب‌سایت کلیک کند
- این کار با قرار دادن وب‌سایت یا دکمه هدف (مثلاً تأیید تراکنش) در یک نامرئی و موقعیت‌دهی آن انجام می‌شود تا کلیک‌های کاربر به عنصر مخفی برخورد کند
- `iframe` ساختاری است که می‌تواند کل یا بخشی از یک صفحه را شامل شود، در هر جای صفحه دیگر قرار گیرد و روی یا زیر فریم‌های دیگر لایه‌بندی شود

#### XSS (cross-site scripting):

- حمله XSS نوعی آسیب‌پذیری امنیتی است که به مهاجم اجازه می‌دهد اسکریپت‌های مخرب را در وب‌سایت‌های معتبر تزریق کند

- وقتی کاربر نظری شامل `<script>` ارسال می‌کند، مرورگر سایر کاربران آن را به عنوان محتوای معتبر اجرا می‌کند، هشدار نمایش داده یا کد مخرب اجرا می‌شود
- XSS ذخیره‌شده: (Persistent XSS)  
 ➤ اسکریپت مخرب روی سرور ذخیره می‌شود (مثلاً در دیتابیس، انجمن یا بخش نظرات)  
 ➤ تمام کاربرانی که صفحه تزریق‌شده را باز می‌کنند تحت تأثیر قرار می‌گیرند

- اگر سرور ورودی را بدون پاک‌سازی بازتاب دهد، اسکریپت مخرب هنگام بازدید کاربر اجرا می‌شود
- XSS بازتابی: (Reflected XSS)  
 ➤ اسکریپت مخرب در URL یا درخواست جاسازی و از سرور بازتاب می‌شود  
 ➤ کاربرانی که روی لینک مخرب کلیک کنند تحت تأثیر قرار می‌گیرند

An attacker crafts a URL:

```
https://example.com/search?q=<script>alert('Hacked!');</script>
```

- If the server echoes back the input without sanitizing it (→vulnerable website), the malicious script executes in the user's browser when they visit the link.

- XSS مبتنی بر: DOM
- > مشابه XSS بازتابی است، اما اسکریپت مخرب از درخواست HTTP فعلی می‌آید
- > آسیب‌پذیری در کد سمت کاربر است و نیاز به تعامل با سرور ندارد
- مثال: وقتی سایت از JavaScript برای پردازش بخش fragment URL استفاده می‌کند، مهاجم می‌تواند URL مخرب بسازد و اسکریپت اجرا شود

Example: A site uses JavaScript to process the URL fragment:

```
document.getElementById("greeting").innerHTML = "Hello, " + location.hash.substring(1);
```

An attacker crafts this URL:

```
https://example.com/page.html#<script>alert('hacked');</script>
```

or this one:

```
https://example.com/page.html#<img src=x onerror="alert('hacked')">
```

How can an attacker trick a victim into clicking the malicious link?

- ایمیل‌های فیشینگ
- پیام‌ها در شبکه‌های اجتماعی
- نظرات در انجمن‌ها یا وبلاگ‌ها
- تبلیغات جعلی
- کوتاه‌کننده‌های URL (مثلاً bit.ly)

:XSS objectives

- سرقت اطلاعات حساس
- ربودن نشست‌های کاربر با سرقت کوکی‌ها (برای جعل هویت کاربر)
- هدایت کاربران به سایت‌های مخرب
- تغییر محتوای وب‌سایت
- نصب بدافزار
- تخریب یا ایجاد محتوای جعلی



## تکنیک های جلوگیری از XSS:

- اعتبارسنجی ورودی: تمام ورودی های کاربر را بررسی کنید تا با فرمت یا نوع مورد انتظار مطابقت داشته باشند، تنها کاراکترهای امن (Whitelisted) را اجازه دهید
  - پاک سازی ورودی: از کتابخانه ها یا فریم ورک ها برای پاک سازی ورودی ها استفاده کنید (مثلاً DOMPurify)
  - کدگذاری خروجی: قبل از نمایش داده ها در وب صفحه، آن ها را کدگذاری کنید تا مرورگر آن ها را به عنوان کد اجرایی تفسیر نکند
- کدگذاری HTML: جایگزینی کاراکترهای ویژه با معادل HTML آن ها:
- **HTML Encoding: replace special characters with their HTML entity equivalents:**
    - ✓ < → &lt;
    - ✓ > → &gt;
    - ✓ " → &quot;
    - ✓ ' → &#x27;
    - ✓ & → &amp;
  - سیاست امنیت محتوا: (CSP)
    - سرور قوانین CSP را در هدرهای پاسخ HTTP به مرورگر می فرستد
    - مرورگر این قوانین را برای تعیین منابع مجاز برای بارگذاری یا اجرا اعمال می کند
    - CSP امکان مشخص کردن منابع معتبر برای اسکریپت ها را با دستور script-src می دهد
    - مثال: Content-Security-Policy: script-src 'self' https://trusted.cdn.com;
    - اجرای اسکریپت ها محدود به همان منبع ('self') یا <https://trusted.cdn.com> می شود
    - به طور پیش فرض، اسکریپت های inline مسدود می شوند مگر با استفاده از دستور 'unsafe-inline' مجاز شوند
    - اسکریپت های تزریق شده مسدود خواهند شد
  - استفاده از HttpOnly برای کوکی های حساس
  - کوکی برای JavaScript سمت کاربر قابل دسترسی نیست
  - این کوکی ها تنها در درخواست های HTTP به سرور ارسال می شوند و نمی توانند توسط اسکریپت ها خوانده، تغییر یا سرقت شوند
  - مثال: Set-Cookie: sessionId=abc123; HttpOnly; Secure;
  - این کوکی از دسترسی یا دستکاری از طریق document.cookie محافظت می شود

## SQL Injection

- بسیاری از برنامه های وب از پایگاه داده SQL برای ذخیره و بازیابی داده ها استفاده می کنند

- SQL Injection از این واقعیت سوءاستفاده می کند که کوئری های SQL به صورت پویا بر اساس ورودی کاربر ساخته می شوند
- اگر ورودی به درستی اعتبارسنجی یا پاک سازی نشود، مهاجم می تواند دستورات SQL را تزریق کند
- این امکان برای مهاجم فراهم می شود که:
  - به داده های حساس دسترسی پیدا کند (مثلاً اطلاعات کاربر، رمز عبور)
  - داده ها را تغییر یا حذف کند
  - عملیات مدیریتی روی پایگاه داده انجام دهد (مثلاً خاموش کردن آن)
  - مکانیزم های احراز هویت را دور بزند
  - احتمالاً کنترل سرور را به دست بگیرد
- گاهی اوقات کوئری های SQL از طریق مرورگر (توسط صفحه وب فرانت اند) ساخته شده و به سرور وب ارسال می شوند

- e.g., the search page of an online book store

```
SELECT * FROM books WHERE category='history';
```

The web page submits this query to the web server.

```
http://www.bookstore.com/find?QUERY=SELECT%20*%20FROM%20books%20WHERE%20category='history';
```

- An attacker can transmit their malicious code to the web server.

```
http://www.bookstore.com/find?QUERY=SELECT%20*%20FROM%20users
```

- اکثر وبسایت ها کوئری های SQL را در سرور بک اند با استفاده از ورودی های کاربر می سازند
- مثال شبه کد سمت سرور:

```

Edit  Copy  ini

    UID = getRequestParamer("UserID");
    Birthday = getRequestParamer("Birthday");
    NID = getRequestParamer("NationalIDNumber");
    Query = "SELECT * FROM users WHERE UserID='" + UID + "' AND Birthday='" + Bir
    Result = SQL_Execute(Query);
  
```

- ورودی های کاربر می توانند از طریق پارامترهای URL یا محتوای درخواست HTTP ارسال شوند

```

UID = getRequestParamter("UserID");
Birthday = getRequestParamter("Birthday");
NID = getRequestParamter("NationalIDNumber");

Query = "SELECT * FROM users WHERE UserID='" + UID + "' AND Birthday='" +
        Birthday + "' AND NationalID='" + NID + "'";

Result = SQL_Execute(Query);

```

If the attacker enters the following input:

- UserID: ' OR '1'='1' --
- Birthday: anything
- NationalIDNumber: anything

➤ The query becomes:

```

SELECT * FROM users WHERE UserID='' OR '1'='1' -- AND Birthday='anything'
AND NationalID=anything';

```

## SQL injection

```

u = getRequestParamter("username");
p = getRequestParamter("password");

Query = "SELECT * FROM users WHERE username='" + u + "' AND password='" +
        p + "'";

Result = SQL_Execute(Query);

```

If the attacker enters the following input:

- username: ' OR ''='
- password: ' OR ''='

➤ The query becomes:

```

SELECT * FROM Users WHERE username='' OR ''='' AND password='' OR ''=''

```

- برای دسترسی به جدول دیگر، مهاجم می‌تواند از دستور UNION SELECT استفاده کند که دو کوئری SELECT نامرتب را ترکیب کرده و داده‌ها را از جداول مختلف بازیابی می‌کند

```
QUERY = "SELECT bookName, bookDescription FROM books WHERE bookID='" +  
inputID + "'" +  
Result = SQL_Execute(Query);
```

Attack scenario:

```
http://www.bookstore.com/book?id=123%20UNION%20SELECT%20username,password%20  
FROM%20users
```

برخی پایگاه‌های داده از اجرای دسته‌ای (Batched) دستورات SQL پشتیبانی می‌کنند

کوئری دسته‌ای شامل دو یا چند دستور SQL است که با سمی‌کالن (;) از هم جدا شده‌اند

مثال:

- Vulnerable server-side code

```
Query = "SELECT * FROM books WHERE category='" + txtCategory + "'";  
Result = SQL_Execute(Query);
```

Malicious input: history'; DROP TABLE users

```
SELECT * FROM books WHERE category='history'; DROP TABLE users
```

Blind sql injection:

- برنامه به SQL Injection آسیب‌پذیر است اما پاسخ‌های HTTP نتایج کوئری SQL را مستقیماً نشان نمی‌دهند
- Blind SQL Injection بر اساس پاسخ‌ها و الگوهای رفتاری سرور عمل می‌کند
- صفحه آسیب‌پذیر بسته به نتیجه عبارت منطقی تزریق‌شده در کوئری SQL تغییر ظاهر می‌دهد

Suppose this query returns book information

```
SELECT bookName, bookDescription FROM books WHERE bookID='123'
```

but it does not return any additional data in the case of an injection

This query returns the book information as well

```
SELECT bookName, bookDescription FROM books WHERE bookID='123' AND '1'='1'
```

but this one does not

```
SELECT bookName, bookDescription FROM books WHERE bookID='123' AND '1'='2'
```

➤ Vulnerable to **boolean-based blind SQL injection**

➤ **Exploiting blind SQL injection**

Attacker's inputs:

```
123' AND SUBSTRING((SELECT password FROM users WHERE username = 'admin'), 1, 1) > 'm'
```

Assuming the previous condition is true:

```
123' AND SUBSTRING((SELECT password FROM users WHERE username = 'admin'), 1, 1) > 't'
```

Assuming the previous condition is false:

```
123' AND SUBSTRING((SELECT password FROM users WHERE username = 'admin'), 1, 1) = 's'
```

If this condition holds true, it confirms that the first character of the password is s.

➤ The attacker can continue this process to determine the full password.

### ➤ Blind SQL injection with cookie!

Consider an application that uses tracking cookies to gather analytics about usage. Requests to the application include a cookie header like this:

```
Cookie: TrackingId=u5YD3PapBcR4lN3e7Tj4
```

When a request containing a TrackingId cookie is processed, the application uses a SQL query to determine whether this is a known user:

```
SELECT TrackingId FROM TrackedUsers WHERE TrackingId = 'u5YD3PapBcR4lN3e7Tj4'
```

This query is vulnerable to SQL injection, but the results from the query are not returned to the user. However, the application does **behave differently** depending on whether the query returns any data.

If you submit a recognized TrackingId, the query returns data and you receive a "Welcome back" message in the response.

- SQL Injection مبتنی بر خطا: (Error-based)
  - بر اساس پیام‌های خطایی است که سرور بازمی‌گرداند
  - یک کوئری نامعتبر می‌تواند خطاهای قابل مشاهده یا عمومی ایجاد کند که سرخ‌هایی درباره ساختار و داده‌های پایگاه داده می‌دهد
- SQL Injection کور مبتنی بر زمان: (Time-based blind)
  - مهاجم کوئری‌ای به پایگاه داده می‌فرستد که باعث توقف (برای چند ثانیه) قبل از پاسخ می‌شود
  - زمان پاسخ پایگاه داده نشان می‌دهد که نتیجه کوئری درست است یا نادرست
- SQL Injection مرتبه اول: (First-order) زمانی رخ می‌دهد که ورودی کاربر از درخواست HTTP به‌طور ناامن در کوئری SQL استفاده شود
- SQL Injection مرتبه دوم: (Second-order) زمانی رخ می‌دهد که برنامه فقط ورودی مستقیم کاربر را بررسی کند، اما داده‌های ذخیره‌شده در سیستم را با سیاست ضعیف‌تر پردازش کند
- در این حالت، ورودی کاربر ابتدا به‌طور ایمن ذخیره می‌شود، اما اگر شامل دستور مخرب SQL باشد، در بخش دیگری از برنامه که حفاظت ندارد اجرا می‌شود

جلوگیری از sql injection:



- اعتبارسنجی و پاک‌سازی ورودی کاربر

- استفاده از کوئری‌های پارامتری (Parameterized Queries)

```
$username = $_POST['username'];  
$password = $_POST['password'];  
  
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?");  
$stmt->bind_param("ss", $username, $password); // "ss" indicates two string parameters  
$stmt->execute();  
$result = $stmt->get_result();
```

- استفاده از Stored Procedures
- رعایت اصل حداقل دسترسی (Least Privilege)
- استفاده از فایروال‌های وب اپلیکیشن (WAFs)



- حمله Directory Traversal (یا Dot-Dot-Slash): مهاجم مسیر فایل‌ها را دستکاری می‌کند تا به دایرکتوری‌ها و فایل‌های خارج از ساختار مجاز وب اپلیکیشن یا سیستم دسترسی پیدا کند
- مسیر عادی:

Edit  Copy 

arduino

```
http://example.com/reports/view?reportfile=report1.txt
```

- مسیر مخرب:

Edit  Copy 

bash

```
http://example.com/files/view?file=../../etc/passwd
```

- این حمله می‌تواند به مهاجم امکان دسترسی به فایل‌های حساس (مثل فایل‌های تنظیمات، اطلاعات ورود یا کد منبع) یا استخراج داده‌های حساس را بدهد

Vulnerable PHP code:

```
<?php
$file = $_GET['reportfile'];
include("files/" . $file);
?>
```

If an attacker supplies file=../../etc/passwd, the script will include the system's password file.

Secure code:

```
<?php
$allowed_files = ['report.txt', 'data.txt'];
$file = $_GET['file'];

if (in_array($file, $allowed_files)) {
    include("files/" . $file);
} else {
    echo "Access denied!";
}
?>
```

## جلوگیری از directory traversal attack:

- اعتبارسنجی ورودی کاربر: فقط نام فایل‌ها یا مسیرهایی که در لیست سفید هستند مجاز باشند
- پاک‌سازی ورودی: حذف هرگونه الگو مانند ../ از ورودی کاربر
- استفاده از مسیرهای مطلق: از مسیرهایی که کاربر وارد می‌کند استفاده نشود؛ مسیرهای مطلق از پیش تعریف شده استفاده شوند
- محدود کردن دسترسی: با تنظیم صحیح سطح دسترسی سیستم فایل، دسترسی به دایرکتوری‌ها و فایل‌های حساس محدود شود
- استفاده از فایروال وب اپلیکیشن (WAF): برای شناسایی و مسدود کردن الگوهای Directory Traversal

اسلاید ششم)

- تمرکز فصل بر شبکه‌های راه‌دور است که کاربر کنترل کمی دارد و ریسک امنیتی بالاست
- سه نوع تهدید اصلی: رهگیری (Interception)، تغییر (Modification)، و وقفه (Interruption)
- معرفی روش‌های مهم برای مقابله با تهدیدهای شبکه‌ای

- Interruption: جلوگیری از دسترسی مجاز
- Modification: تغییر غیرمجاز
- Interception: مشاهده غیرمجاز



- وقتی داده‌ها از محیط محافظت شده خارج می‌شوند، افراد در مسیر می‌توانند آن‌ها را مشاهده یا رهگیری کنند
- رهگیری سیگنال یک آسیب‌پذیری جدی شبکه‌ای است
- ارتباطات داده چه از طریق سیمی و چه بی‌سیم منتقل شوند، در هر دو حالت در معرض حمله هستند (با درجات متفاوت از سهولت حمله)

- ساده‌ترین اتصال شبکه یک سیم به نام کابل اترنت است
- Packet Sniffer: دستگاهی که تمام بسته‌های موجود در LAN را دریافت می‌کند
- همچنین می‌توان کارت شبکه را دوباره برنامه‌ریزی کرد تا آدرس منحصر به فرد کارت دیگری در LAN را جعل کند

- **Cable Splicing (جاسوسی از کابل):** مهاجم می‌تواند با بریدن کابل و وصل کردن یک کابل ثانویه، یک کپی از همه سیگنال‌های در حال عبور دریافت کند.
- روش دیگر این است که مهاجم بدون بریدن کابل، قسمتی از رسانای داخل کابل را نمایان کرده و به آن وصل شود تا داده‌ها را شنود کند.

- **Radiation (تشعشع الکترومغناطیسی):** سیم‌ها هنگام انتقال داده، مقداری امواج الکترومغناطیسی از خود منتشر می‌کنند.
- **Inductance (القای الکترومغناطیسی):** مهاجم می‌تواند بدون تماس مستقیم با سیم، با نزدیک کردن یک آنتن به کابل این تشعشعات را دریافت کند.
- این کار به او امکان می‌دهد سیگنال‌های عبوری از کابل را شنود کند، حتی بدون اینکه کابل را قطع یا دستکاری کند.

❓ **فیبر نوری** داده‌ها رو به صورت نور منتقل می‌کنه، نه جریان الکتریکی.

❓ برای همین:

- شنود از طریق **القای الکترومغناطیسی** (مثل کابل مسی) غیرممکنه.
- **Splicing (بریدن و وصل کردن)** (فیبر خیلی سخت‌تره و معمولاً باعث افت انرژی و کیفیت سیگنال می‌شه، بنابراین احتمال تشخیص سریع بالاست).

❓ به همین خاطر فیبر نوری در مقایسه با کابل مسی، از نظر امنیت بسیار مقاوم‌تر در برابر شنود محسوب می‌شه.

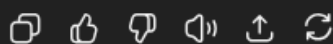
- سیگنال‌های بی‌سیم در هوا منتقل می‌شن → ذاتاً آسیب‌پذیرترین.
- سیگنال خیلی در معرض دید قرار داره.
- مهم‌ترین نوع ارتباط بی‌سیم در امنیت کامپیوتر: **WiFi** :

- همه‌ی لینک‌های ارتباطی بین نودهای شبکه ممکنه شکسته بشن.
- کاربرها معمولاً کنترلی روی مسیر سیگنال ندارن.
- برای حفظ محرمانگی ارتباطات از رمزنگاری استفاده می‌شه.

- تهدید: تغییر داده‌ها هنگام انتقال
- تغییر داده‌ها در مسیر
- درج محتوای جدید
- تکرار یک ارتباط قبلی (Replay)

- رسانه‌های ارتباطی آسیب‌پذیر به فساد داده هستند
- پروتکل‌های ارتباطی ویژگی‌هایی برای بررسی و اصلاح برخی خطاهای انتقال دارند
- TCP/IP تضمین می‌کند که گیرنده داده‌ها را کامل، صحیح و مرتب دریافت کند، حتی در صورت وجود خطا در انتقال

- این مکانیزم‌ها کافی نیستند برای مقابله با فساد داده ناشی از حمله مخرب



چرا؟

چون مکانیزم‌های معمول مثل بررسی خطا یا تصحیح خطا فقط خطاهای تصادفی در انتقال را شناسایی و اصلاح می‌کنن، ولی حملات مخرب عمدی هستند و ممکنه داده‌ها به‌طور هدفمند تغییر داده شوند یا دستکاری پیچیده‌ای انجام بشه که این مکانیزم‌ها قادر به شناسایی یا جلوگیری از آن نباشند.

- حمله ترتیب‌دهی: (Sequencing attack) جابجایی ترتیب داده‌ها
- مثال: بسته دوم قبل از بسته اول به گیرنده می‌رسد
- خطاهای ترتیب‌دهی در ترافیک شبکه رایج هستند
- پروتکل‌های شبکه مثل TCP ترتیب درست ترافیک را تضمین می‌کنند
- برنامه‌های کاربردی همیشه مشکلات ترتیب‌دهی در جریان داده را شناسایی یا اصلاح نمی‌کنند

- حمله جایگزینی: (Substitution attack) جایگزینی یک بخش از جریان داده با بخش دیگر
- مهاجم می‌تواند با چسباندن بخشی از یک ارتباط به ارتباط دیگر این حمله را انجام دهد
- راهکارها:

➤ رمزنگاری کل پیام

➤ ایجاد بررسی یکپارچگی (Integrity check)

- حمله درج: (Insertion attack) وارد کردن مقادیر جدید در جریان داده، شبیه جایگزینی
- برای درج داده، حتی نیازی به شکستن رمزنگاری نیست اگر مهاجم دقیقاً بداند کجا داده رمزگذاری شده را وارد کند

- حمله تکرار: (Replay attack) مهاجم یک انتقال داده معتبر را رهگیری و بعداً دوباره ارسال می‌کند تا گیرنده اقدامات غیرمجاز انجام دهد
- مثال: تراکنش‌های مالی
- رهگیر نیازی به دانستن محتوای پیام یا فرمت آن ندارد
- مقابله با حمله تکرار: استفاده از شماره ترتیب
- ➤ گیرنده آخرین شماره پیام دریافت‌شده را ذخیره و پیام‌های جدید را بررسی می‌کند که شماره آن‌ها بزرگ‌تر از پیام قبلی باشد

- حمله تکرار فیزیکی: (Physical replay attack) نوعی حمله تکرار در سیستم‌های مبتنی بر سیگنال یا دستگاه‌های فیزیکی
- مهاجم سیگنال‌های فیزیکی (مثل فرکانس رادیویی، تصاویر یا ورودی‌های حسی دیگر) را ضبط و دوباره پخش می‌کند تا سیستم را فریب دهد
- مثال: احراز هویت بیومتریک

- آخرین کلاس حملات شبکه مربوط به دسترس‌پذیری (Availability) است، سومین رکن مثلث C-I-A
- شبکه‌ها و اینترنت خدمات پایدار دارند؛ از ابتدا افزونه‌پذیری و تحمل خطا مهم بوده و همچنان پابرجاست

- برخلاف نقض محرمانگی و یکپارچگی، حمله انکار سرویس (DoS) دو حالتی نیست
  - ظرفیت سرویس می‌تواند کاهش یابد
- مثال: اطلاعات نادرست که جداول مسیریابی رو مسموم می‌کنند و یک دامنه را غیرقابل دسترس می‌کنند

- مشکل مهاجم: دانستن اینکه کدام حملات را روی کدام دستگاه‌ها اجرا کند
- دستگاه‌های موجود در شبکه
- توپولوژی شبکه
- سرویس‌های فعال روی هر دستگاه
- روش ساده برای جمع‌آوری اطلاعات شبکه: استفاده از پویشگر پورت (Port Scanner)
- برنامه‌ای که برای یک آدرس IP مشخص، پورت‌های پاسخ‌دهنده و آسیب‌پذیری‌های شناخته‌شده را گزارش می‌دهد
- مثال Nmap، netcat

- Scanning معمولاً اولین مرحله در یک حمله است

- Nmap پورت‌ها را بررسی می‌کند تا سرویس‌های پاسخ‌دهنده را شناسایی کند

```
Nmap scan report
192.168.1.1 / somehost.com (online) ping results
address: 192.168.1.1 (ipv4)
hostnames: somehost.com (user)
The 83 ports scanned but not shown below are in state: closed


| Port | State | Service  | Reason  | Product     | Version  | Extra info     |
|------|-------|----------|---------|-------------|----------|----------------|
| 21   | tcp   | open     | ftp     | syn-ack     | ProFTPD  | 1.3.1          |
| 22   | tcp   | filtered | ssh     | no-response |          |                |
| 25   | tcp   | filtered | smtp    | no-response |          |                |
| 80   | tcp   | open     | http    | syn-ack     | Apache   | 8.2.3 (CentOS) |
| 106  | tcp   | open     | pop3pw  | syn-ack     | poppassd |                |
| 110  | tcp   | open     | pop3    | syn-ack     | pop3d    |                |
| 111  | tcp   | filtered | rpcbind | no-response |          |                |
| 113  | tcp   | filtered | auth    | no-response |          |                |
| 143  | tcp   | open     | imap    | syn-ack     | imapd    | 17.3           |
| 443  | tcp   | open     | http    | syn-ack     | Apache   | 8.2.3 (CentOS) |
| 465  | tcp   | open     | unknown | syn-ack     |          |                |
| 646  | tcp   | filtered | ldp     | no-response |          |                |
| 993  | tcp   | open     | imap    | syn-ack     | imapd    | 17.3           |
| 995  | tcp   | open     |         | syn-ack     |          |                |
| 2049 | tcp   | filtered | nfs     | no-response |          |                |
| 3306 | tcp   | open     | mysql   | syn-ack     | MySQL    | 5.0.45         |
| 8443 | tcp   | open     | unknown | syn-ack     |          |                |


34 sec. scanned
1 host(s) scanned
1 host(s) online
0 host(s) offline
```

- Port scanning به مهاجم سه چیز می‌گوید:
- پورت‌ها یا سرویس‌های فعال و پاسخ‌دهنده در سیستم هدف

➤ سیستم عامل نصب شده روی سیستم هدف  
➤ برنامه ها و نسخه های آنها

- مهاجم می تواند **connectivity** شبکه را نیز شناسایی کند
- با **scan** کردن کل **subnet** ، مثال: 192.168..

```
Nmap scan report for router (192.168.1.1)
Host is up (0.00s latency).
MAC Address: 00:11:22:33:44:55 (Brand 1)

Nmap scan report for computer (192.168.1.39)
Host is up (0.78s latency).
MAC Address: 00:22:33:44:55:66 (Brand 2)

Nmap scan report computer (192.168.1.43)
Host is up (0.010s latency).
MAC Address: 00:11:33:55:77:99 (Brand 3)

Nmap scan report for unknown device 192.168.1.44
Host is up (0.010s latency).
MAC Address: 00:12:34:56:78:9A (Brand 4)

Nmap scan report for computer (192.168.1.47)
Host is up.
```

---

- مدیران شبکه یا مالکان سیستم نیز از این ابزارها برای بررسی شبکه استفاده می کنند  
➤ گزارش می دهد کدام دستگاه ها نرم افزار قدیمی و آسیب پذیر دارند یا کدام پورت ها به طور غیرضروری باز هستند  
➤ مدیران شبکه های بزرگ از اسکنر برای مستندسازی و بررسی همه دستگاه های متصل به شبکه استفاده می کنند

اسلاید هفتم)

- Denial of Service (DoS) attack: تلاش مخرب برای اختلال در عملکرد سیستم، سرویس یا شبکه
- هدف: غیرقابل دسترس کردن هدف برای کاربران قانونی  
➤ کاربران از دسترسی به سرویس ها یا داده های مجاز محروم می شوند
- تمرکز روی **availability** است، نه داده ها  
➤ محرمانگی و یکپارچگی جلوگیری از دسترسی غیرمجاز را شامل می شوند، دسترسی پذیری حفظ دسترسی مجاز را هدف دارد

- نگرانی اصلی برای کسب و کارها و زیرساخت‌های حیاتی

- محرمانگی و یکپارچگی دو حالتی هستند: داده‌ها یا خصوصی و دست نخورده‌اند یا نیستند
- دسترسی پذیری می‌تواند پیچیده‌تر باشد: سرویس وجود دارد ولی با کمبود ظرفیت یا پاسخ‌دهی نامطلوب
- DoS می‌تواند از کامل عدم دسترسی تا کندی محسوس یا مزاحمت جزئی متغیر باشد

- DoS attacks چگونه کار می‌کنند:

- **Overloading resources:** پر کردن هدف با ترافیک تا از کار بیفتد
- **Exploiting vulnerabilities:** سوءاستفاده از آسیب‌پذیری‌ها برای کرش کردن سیستم
- **Blocking access:** قطع یا غیرفعال کردن لینک ارتباطی بین دو نقطه

- انواع: DoS attacks

- **Volumetric attacks:** پر کردن شبکه با حجم زیادی داده (مثال UDP Flood، ICMP Flood)
- **Protocol attacks:** سوءاستفاده از ضعف‌های پروتکل شبکه (مثال SYN Flood، Smurf Attack)
- اختلال در مکانیزم‌های مسیریابی شبکه و جلوگیری از رسیدن درخواست‌ها به سرور
- **Application layer attacks:** هدف قرار دادن برنامه‌ها یا سرویس‌ها (مثال: HTTP Flood، Email Flood)
- سوءاستفاده از آسیب‌پذیری برنامه و کرش کردن آن
- دستکاری داده‌های کنترل دسترسی، حذف مجوزها یا غیرفعال کردن مکانیزم کنترل دسترسی

❓ رایج‌ترین نوع حمله DoS همون **Flooding** هست.

❓ مهاجم سعی می‌کنه سرور رو با ارسال دستورات سریع‌تر از توان پردازش سرور پر کنه.  
➤ سرورها معمولاً وقتی زیاد شلوغه، درخواست‌ها رو توی صف نگه می‌دارن تا بعد پردازش کنن.  
➤ اما اگه دستورات با سرعت ادامه داشته باشه، سرور فضای کافی برای نگه داشتن درخواست‌ها نداره و کار می‌افته.

Flooding ❓ خودش ساده است، ولی روش انجامش می‌تونه پیچیده باشه.  
➤ اگه پهنای باند مهاجم بیشتر از سرور باشه، با **تفاوت منابع** می‌تونه سرور رو کاملاً از کار بندازه.

❓ در واقع هر سروری همیشه در معرض تهدید مهاجمی با منابع بیشتر هست.

- دستور **ping (ICMP)** برای تست اتصال بین دو دستگاه استفاده می‌شود
- معمولاً اندازه بسته کوچک است (مثلاً 32 یا 64 بایت)
- در حمله **Ping of Death**، مهاجم بسته ICMP بزرگ‌تر از حد مجاز (65,535 بایت) می‌فرستد
- ➤ بسته بزرگ در حین انتقال به قطعات کوچک‌تر تقسیم می‌شود
- ➤ سیستم هدف تلاش می‌کند قطعات را به بسته اصلی بازسازیشده ترکیب کند
- ➤ سیستم‌های قدیمی با فرآیند بازسازی ضعیف ممکن است **باگ حافظه (buffer overflow)** داشته باشند یا کرش کنند
- حمله اصلی **Ping of Death** امروزه کمتر رایج است
- نوع مرتبط و رایج‌تر **ICMP flood attack** :
- تو **ICMP flooding**، مهاجم کلی بسته **ICMP Echo Request** می‌فرسته به سمت سرور.
- سرور هم سعی می‌کند به همه این درخواست‌ها جواب بده (**Echo Reply**) و این باعث می‌شه **CPU**، **حافظه** و **پهنای باند**ش پر بشه.
- آگه تعداد بسته‌ها خیلی زیاد باشه، سرور **گیر می‌کنه** و نمی‌تونه ترافیک واقعی رو هندل کنه.
- سرعت حمله محدود می‌شه به **کمترین پهنای باند مسیر حمله**.
- تو **Smurf attack**، مهاجم از آدرس **broadcast** استفاده می‌کنه تا پیام به همه دستگاه‌های شبکه بره) مثلاً 192.168.1.255 به همه دستگاه‌های 192.168.1.x.
- مهاجم **آدرس منبع بسته ping** رو جعل می‌کنه تا شبیه قربانی باشه
- همه دستگاه‌ها بسته **ICMP Echo Request** رو دریافت می‌کنن و جواب (**Echo Reply**) می‌فرستن به آدرس جعلی قربانی
- در نتیجه، کل زیرشبکه برای **ضرب کردن حمله** استفاده می‌شه و اثرش چند برابر می‌شه
- تو **SYN Flood**، مهاجم از **TCP handshake** سوءاستفاده می‌کنه تا سرور رو گیج کنه
- گاهی بسته‌ها تو مسیر **گم** یا **خراب** می‌شن
- ➤ سرور به صف به نام **SYN\_RCV** داره که پیگیری می‌کنه برای چه اتصال‌هایی **SYN-ACK** فرستاده شده ولی جواب **ACK** هنوز نیومده
- مهاجم کلی **SYN packet** می‌فرسته به سرور

- سرور به هر SYN جواب **SYN-ACK** می‌دهد و منابع برای اتصال اختصاص می‌دهد، اتصال رو تو صف **SYN\_RCV** قرار می‌دهد و منتظر **ACK** می‌مونه
- مهاجم هیچ وقت **ACK** نمی‌فرسته، در نتیجه اتصال نیمه‌باز می‌مونه
- سرور این اتصالات نیمه‌باز رو نگه می‌داره و **حافظه و CPU** مصرف می‌کنه
- وقتی صف پر بشه، سرور نمی‌تونه **اتصال جدید قبول کنه** و کاربران واقعی دچار **Denial of Service** می‌شن

- مهاجم‌ها معمولاً به کار دیگه هم می‌کنن: **آدرس برگشتی جعلی** تو SYN اول می‌ذارن
- دلیل‌ها:
  - نمی‌خوان **آدرس واقعی** شون لو بره اگه کسی بسته‌ها رو تو صف SYN\_RCV بررسی کنه
  - می‌خوان بسته‌های مخرب مثل **SYN واقعی** به نظر برسند و سرور فکر کنه اتصال واقعی ست

- **DNS spoofing**: وقتی مهاجم روند پرس‌وجوی DNS رو **دستکاری** یا **رهگیری** می‌کنه و به جای IP واقعی، IP جعلی می‌ده
- **DNS Cache Poisoning**: نفوذ به resolver و اضافه کردن رکورد جعلی به کش (راه حل) **DNSSEC** :
- **Man-in-the-Middle**: رهگیری پرس‌وجوی DNS بین کاربر و resolver یا بین دو سرور و جواب دادن با پاسخ جعلی
- **Compromising Authoritative DNS Servers**: هک کردن سرور DNS اصلی و تغییر رکوردها (راه حل) **DNSSEC** :
- **Local DNS Spoofing**: تغییر تنظیمات DNS روی دستگاه کاربر (مثلاً با بدافزار) تا به سرور DNS مخرب وصل شود

- به روش دیگه برای ایجاد **Denial of Service** با مشکل در **Address Resolution** اینه که خود سیستم DNS اینترنت رو از کار بندازیم
- 2002: حمله flood بزرگ به **top-level domain DNS servers**
- 2007: حمله flood بزرگ به **root DNS servers**
- ✓ اثر این حمله‌ها با **anycast technology** خیلی کم شد

- هر روتر اینترنت زیرشبکه خودش و همسایه‌ها رو با **BGP** اعلام می‌کنه
- **BGP** پروتکل مبتنی بر اعتماد است و به‌طور پیش‌فرض اعلان‌های مسیر را احراز هویت نمی‌کنه



- هیچ اعتبارسنجی داخلی وجود ندارد؛ BGP بررسی نمی‌کند مسیرهای اعلام‌شده واقعی باشند
  - >یه اشتباه کوچک یا به‌روزرسانی مخرب می‌تونه کل اینترنت رو تحت تاثیر قرار بده
  - دیتابیس‌ها یا بافرهایی که پیام یا ایمیل ذخیره می‌کنن، می‌تونن با ورود ناگهانی تعداد زیادی پیام پر بشن
  - فایل‌های لاگ هم می‌تونن با اجرای مکرر یه عمل که لاگ ثبت می‌کنه پر بشن
  - حتی سیستم‌های احراز هویت هم ممکنه در معرض DOS باشن
  - >بعضی سرویس‌ها بعد از چند بار ورود ناموفق حساب رو موقت یا دائم قفل می‌کنن
  - >مهاجم می‌تونه با ورود مکرر ناموفق، دسترسی واقعی کاربر رو مسدود کنه
- 
- **DDoS attack** یعنی استفاده از چندین منبع — گاهی هزاران یا میلیون‌ها دستگاه آلوده — برای سیل کردن هدف با ترافیک یا درخواست‌ها
  - **Trojan** روی دستگاه‌های زیادی نصب می‌کنه؛ هر کدوم از این سیستم‌ها می‌شن **bot** یا **zombie**
  - مهاجم یه قربانی انتخاب می‌کنه و به زامبی‌ها دستور حمله می‌ده
  - نه دستگاه‌ها و نه صاحبانشون نمی‌دونن بخشی از حمله هستن
  - هیچ سازمان یا هاست در دسترس از حمله مصون نیست
- 
- **Zombies:** دستگاه‌هایی که تکه‌هایی از کد مخرب رو تحت کنترل از راه دور اجرا می‌کنن
  - چون معمولاً به کاربر آسیبی نمی‌رسونن (فقط منابع سیستم و شبکه مصرف می‌کنن)، اغلب تشخیص داده نمی‌شن
  - **Botnets:** شبکه‌ای از bot ها که برای حملات گسترده **Dos** استفاده می‌شن، از چندین سایت به‌طور موازی علیه قربانی
  - >مثال: شبکه (2009) Koobface
  - کسانی که دستگاه‌ها رو به bot تبدیل می‌کنن **botmaster** نام دارن
  - >گاهی botmaster ها **botnet** خودشون رو اجاره می‌دن به دیگران
- 
- برای مقابله با **DDoS attacks** مدیران می‌تونن از روش‌های معمول **Dos** استفاده کنن:
  - **Tuning:** >تنظیم تعداد سرورهای فعال
  - **Load balancing:** >توزیع مساوی بار بین سرورهای موجود
  - **Shunning:** >کاهش سرویس به ترافیک از بازه‌های آدرس مشخص
  - **Blacklisting:** >رد کردن اتصال از آدرس‌های خاص
  - همین روش‌ها برای **DDoS** هم استفاده می‌شن، ولی در مقیاس بزرگ‌تر و در لبه شبکه اعمال می‌شن

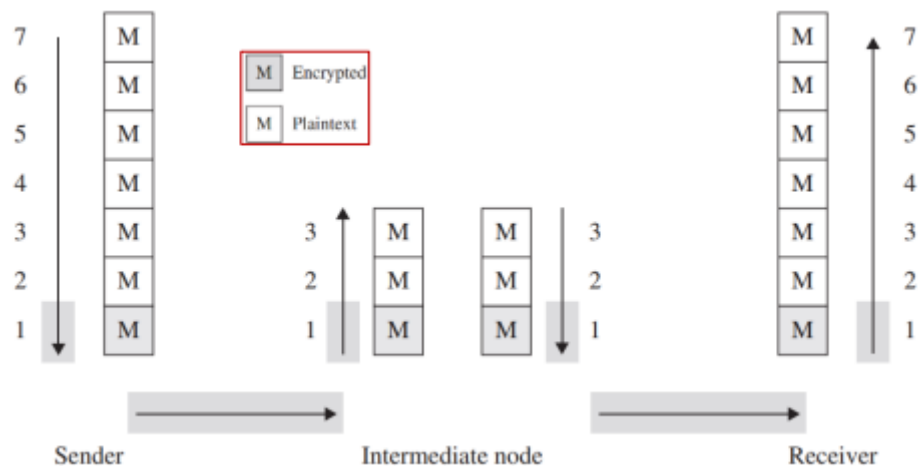
- تا اینجا: تهدیدها و حملات در شبکه
- باقی فصل به سه دسته کنترل می‌پردازد:
- **Encryption:** توضیح ساختاری نحوه استفاده از رمزنگاری و معرفی دو کاربرد شبکه‌ای :  
VPN و SSL
- **Firewall**

➤ **Intrusion detection/protection system**

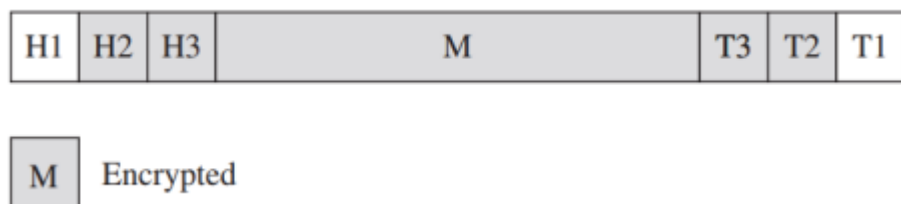
- قبل از بررسی استفاده از رمزنگاری برای مقابله با تهدیدهای شبکه چند نکته:
  - رمزنگاری فقط چیزهایی که رمز شده‌اند رو محافظت می‌کنه
  - طراحی الگوریتم‌های رمزنگاری بهتره به متخصص‌ها سپرده بشه
  - امنیت رمزنگاری به مدیریت کلیدها وابسته است
  - رمزنگاری معجزه نیست؛ سیستم با طراحی ضعیف حتی با رمزنگاری هم ضعیف باقی می‌مونه

- تو شبکه، رمزنگاری می‌تونه روی دو حالت انجام بشه:
- **Link encryption:** ➤ بین دو میزبان
- **End-to-end encryption:** ➤ بین دو برنامه
- هر کدوم کارکرد، نقاط قوت و ضعف خودشون رو دارن

- **Link encryption:** داده‌ها درست قبل از قرار گرفتن روی لینک فیزیکی رمزگذاری می‌شن
- رمزگذاری در لایه 1 یا 2 OSI انجام می‌شه
- رمزگشایی درست وقتی داده به سیستم دریافت‌کننده می‌رسه انجام می‌شه
- **Link encryption** ➤ ارتباط رو از یک گره به گره بعدی محافظت می‌کنه

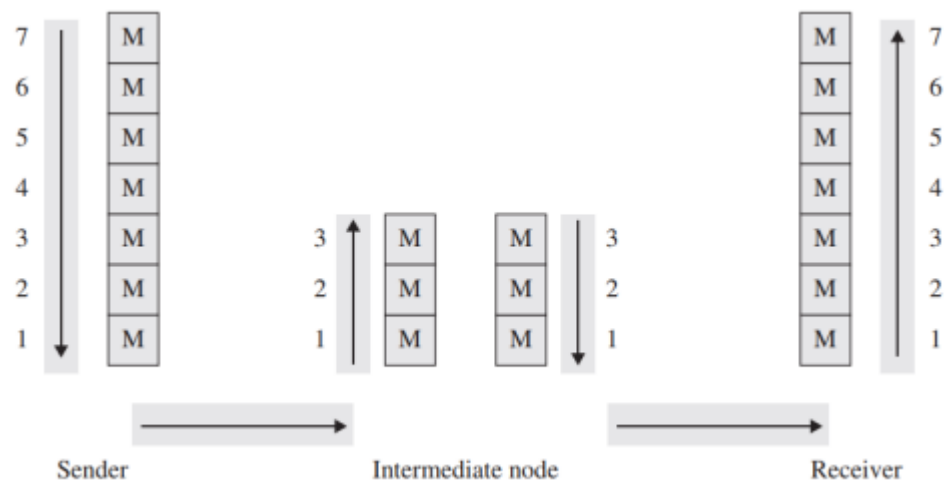


- رمزنگاری پیام رو در حین انتقال بین دو کامپیوتر محافظت می‌کنه، اما داخل خود میزبان‌ها پیام متن ساده‌ه  
 ➤ چون رمزنگاری در پایین‌ترین لایه پروتکل اضافه می‌شه، پیام در تمام لایه‌های دیگه فرستنده و گیرنده قابل مشاهده است

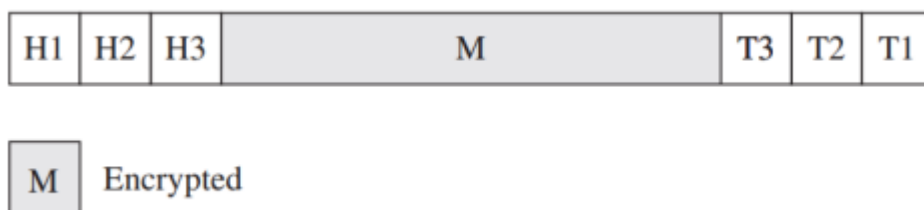


- دستگاه‌های سخت‌افزاری رمزنگاری سریع و قابل اعتماد عمل می‌کنن  
 ➤ در این حالت، **link encryption** برای سیستم عامل و کاربر قابل مشاهده نیست
- Link encryption وقتی مناسب‌تره که خط انتقال آسیب‌پذیرترین نقطه باشه
- اگر همه میزبان‌ها نسبتاً امن باشن ولی رسانه ارتباطی مشترک یا ناامن باشه، link encryption کنترل ساده‌ایه برای استفاده

- **End-to-end encryption:** امنیت رو از مبدا تا مقصد فراهم می‌کنه
- رمزنگاری در لایه‌های بالای OSI معمولاً لایه 7، گاهی 5 یا 6 انجام می‌شه
- End-to-end encryption ➤ ارتباط رو از فرستنده تا گیرنده پوشش می‌ده  
 ➤ حتی اگر پیام از گره‌های ناامن عبور کنه، در حین انتقال محافظت شده و قابل افشا نیست



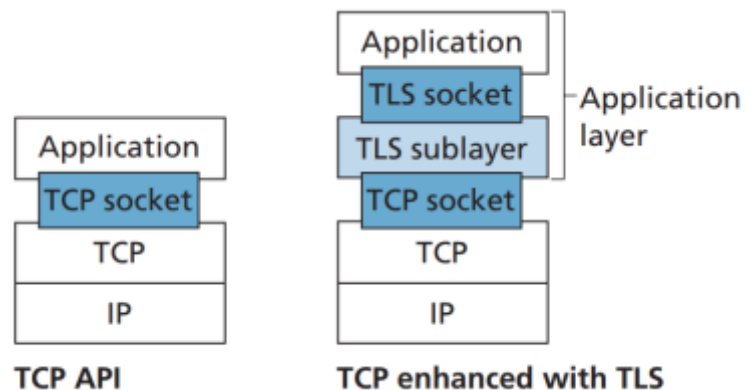
- چون رمزنگاری قبل از همه پردازش‌های مسیر و انتقال لایه‌ها انجام می‌شود، پیام در کل شبکه به صورت رمزگذاری شده منتقل می‌شود
- فقط قسمت داده‌ای پیام محافظت می‌شود، معمولاً هدرها حساس نیستند
- هر جفت کاربر باید یک کلید رمزنگاری منحصر به فرد داشته باشند
- میزبان‌های میانی نیازی به رمزگذاری یا رمزگشایی ندارند و بنابراین تجهیزات رمزنگاری برایشان لازم نیست



TLS = transport layer security

- اگر رمزنگاری (محرمانگی) نباشد، نفوذگر می‌تونه سفارش Bob و اطلاعات کارت پرداختش رو رهگیری و بدزده
- اگر یکپارچگی داده رعایت نشود، نفوذگر می‌تونه سفارش یا آدرس Bob رو تغییر بده
- اگر احراز هویت سرور نباشد، سرور می‌تونه لوگوی Alice Incorporated رو نشون بده ولی در واقع سایت تحت کنترل Trudy باشه
- Trudy > می‌تونه پول Bob رو بدزده یا > با جمع‌آوری نام، آدرس و کارت اعتباری Bob، سرقت هویت انجام بده
- TLS قابلیت‌های رمزنگاری، یکپارچگی داده، احراز هویت سرور و احراز هویت کلاینت رو به TCP اضافه می‌کنه
- نسخه قدیمی‌تر این پروتکل (SSL (Secure Sockets Layer بود
- SSL توسط Netscape در دهه 1990 برای محافظت از ارتباط بین مرورگر و سرور طراحی شد
- SSL 2.0 (1995) > ، SSL 3.0 (1996)

- در 1999، IETF نسخه SSL 3.0 رو بهبود داد و نامش رو گذاشت **TLS**
- (2006) TLS 1.1، (2008) TLS 1.2، (2018) TLS 1.3
- اصطلاح SSL اغلب برای هر دو مجموعه پروتکل **SSL** و **TLS** استفاده می‌شه
- TLS توسط تمام مرورگرها و سرورهای وب محبوب پشتیبانی می‌شه
- TLS اغلب برای امن کردن تراکنش‌های **HTTP** استفاده می‌شه
- چون **TCP** رو امن می‌کنه، می‌تونه توسط هر برنامه‌ای که روی **TCP** اجرا می‌شه هم استفاده بشه
- TLS یک رابط برنامه‌نویسی ساده (**API**) با سوکت‌ها فراهم می‌کنه، مشابه **API** خود **TCP**
- وقتی برنامه‌ای بخواد از **TLS** استفاده کنه، باید کلاس‌ها یا کتابخانه‌های **SSL** رو اضافه کنه



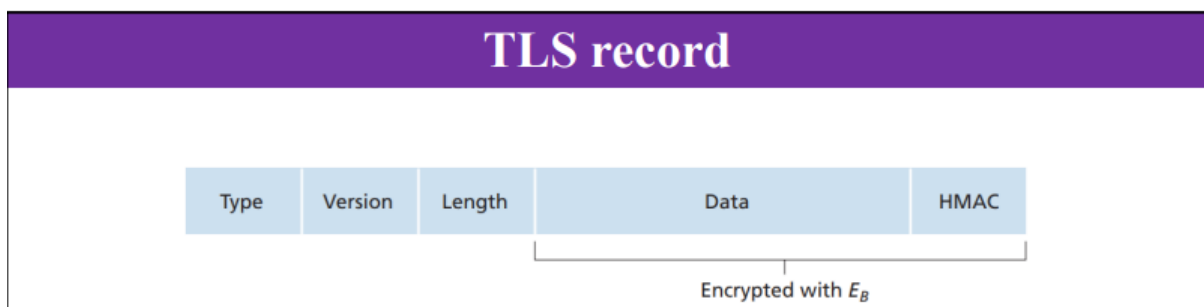
- برای فهم کلی **TLS**، ابتدا یک نسخه ساده‌شده به نام **“almost-TLS”** معرفی می‌کنیم
- بعد از توضیح **almost-TLS**، **TLS** واقعی شرح داده می‌شه
- **TLS** سه فاز داره **handshake**؛ استخراج کلید، انتقال داده
- مثال: ارتباط بین یک کلاینت (**Bob**) و یک سرور (**Alice**) که کلید خصوصی/عمومی و گواهی داره که هویتش رو به کلید عمومی وصل می‌کنه
- در فاز **handshake**، **Bob** باید:
  - یک اتصال **TCP** با **Alice** برقرار کنه
  - مطمئن بشه که **Alice** واقعاً همونه
  - یک کلید محرمانه اصلی (**master secret key**) برای **Alice** بفرسته
- این کلید محرمانه توسط هر دو برای تولید تمام کلیدهای متقارن مورد نیاز جلسه **TLS** استفاده می‌شه

## Almost-TLS: Handshake

- بعد از برقراری اتصال TCP ، Bob یک پیام **hello** به Alice می‌فرسته
  - Alice با گواهی‌نامه خودش پاسخ می‌ده که شامل **کلید عمومی** شه
  - چون گواهی توسط **CA** تأیید شده، Bob مطمئننه که کلید عمومی واقعاً متعلق به Alice ه
  - Bob یک **Master Secret (MS)** تولید می‌کنه، اون رو با **کلید عمومی Alice** رمزگذاری می‌کنه تا **Encrypted Master Secret (EMS)** ساخته بشه و به Alice می‌فرسته
  - Alice با **کلید خصوصی** خودش **EMS** رو رمزگشایی می‌کنه تا **MS** رو بدست بياره
  - بعد از این فاز، فقط **Alice** و **Bob** از **master secret** این جلسه TLS خبر دارن
  - در اصل، **MS** می‌تونه به عنوان **کلید متقارن** جلسه برای رمزنگاری و بررسی یکپارچگی داده‌ها استفاده بشه
  - اما امن‌تره که **Alice** و **Bob** **کلیدهای متفاوتی** برای رمزنگاری و یکپارچگی داده داشته باشن
  - بنابراین، **MS** برای تولید **چهار کلید** استفاده می‌شه:
    - **EB**: کلید رمزنگاری برای داده‌های Alice → Bob
    - **MB**: کلید HMAC برای داده‌های Alice → Bob
    - **EA**: کلید رمزنگاری برای داده‌های Bob → Alice
    - **MA**: کلید HMAC برای داده‌های Bob → Alice
  - این کار می‌تونه با تقسیم **MS** به **چهار کلید** انجام بشه، ولی در TLS واقعی کمی پیچیده‌تره
- 
- حالا که **Alice** و **Bob** **چهار کلید** جلسه **مشترک** دارن، می‌تونن داده‌های امن رو از طریق اتصال TCP رد و بدل کنن
  - چون TCP یک **پروتکل جریان بایت‌ه**، رویکرد طبیعی اینه که TLS داده‌های برنامه رو **بلادرنگ رمزنگاری** کنه و به TCP بده
  - سوال **HMAC**: بررسی یکپارچگی داده‌ها کجا گذاشته بشه؟
    - نمی‌خوایم تا پایان جلسه TCP صبر کنیم و بعد یکپارچگی تمام داده‌های Bob رو بررسی کنیم
- 
- TLS جریان داده‌ها رو به **رکورد تقسیم** می‌کنه
  - به هر رکورد یک **HMAC** برای بررسی یکپارچگی اضافه می‌کنه
  - بعد رکورد **HMAC +** رو رمزنگاری می‌کنه
  - این بسته رمزنگاری شده بعداً به **TCP** برای انتقال روی اینترنت فرستاده می‌شه

آیا almost-TLS امنه؟

- Trudy ممکنه دو بخش داده‌ای که Bob فرستاده رو بگیره، ترتیبشون رو برگردونه، شماره‌های ترتیب TCP رو تنظیم کنه و بعد به Alice بفرسته
- راه حل:
  - Bob یک شمارنده شماره ترتیب نگه می‌داره که از صفر شروع می‌شه و برای هر رکورد TLS افزایش پیدا می‌کنه
  - شماره ترتیب داخل رکورد فرستاده نمی‌شه
  - هنگام محاسبه HMAC ، Bob شماره ترتیب رو هم لحاظ می‌کنه  $\rightarrow \text{HMAC} = \text{hash}(\text{data} + \text{MB} + \text{current\_sequence\_number})$
  - Alice شماره ترتیب Bob رو پیگیری می‌کنه



- سه فیلد اول رمزنگاری نمی‌شن
- فیلد **type** مشخص می‌کنه که رکورد پیام **handshake** ، داده برنامه، یا رکورد بستن اتصال TLS ه
- TLS در سمت گیرنده از فیلد **length** استفاده می‌کنه تا رکوردها رو از جریان بایت TCP ورودی استخراج کنه

- SSL الزامی به استفاده از الگوریتم کلید متقارن یا کلید عمومی خاص نداره
- TLS اجازه می‌ده Alice و Bob در فاز **handshake** روی الگوریتم‌های رمزنگاری توافق کنن
- در فاز **handshake** ، Alice و Bob **nonce** ها رو رد و بدل می‌کنن که برای ساخت کلیدهای جلسه (EB) ، EA ، MB ، MA استفاده می‌شن

Real TLS handshake:

1. کلاینت لیستی از الگوریتم‌های رمزنگاری قابل پشتیبانی خودش رو همراه با **nonce** کلاینت می‌فرسته (پیام Client\_Hello)
2. سرور از لیست یک الگوریتم متقارن (مثلاً AES) ، یک الگوریتم کلید عمومی (مثلاً RSA با طول کلید مشخص (و الگوریتم MD5) HMAC یا (SHA-1) انتخاب می‌کنه
- سرور این انتخاب‌ها، گواهی‌نامه و **nonce** سرور رو برمی‌گردونه (پیام Server\_Hello)

3. کلاینت گواهی نامه رو بررسی می کنه، کلید عمومی سرور رو استخراج می کنه  
 • یک **Pre-Master Secret (PMS)** تولید می کنه، اون رو با کلید عمومی سرور رمزنگاری می کنه و PMS رمزنگاری شده رو به سرور می فرسته
4. با استفاده از همان تابع ( **key derivation** ) طبق استاندارد (TLS) ، کلاینت و سرور به طور مستقل **Master Secret (MS)** رو از PMS و nonce ها محاسبه می کنن  
 • MS سپس تقسیم می شه تا دو کلید رمزنگاری و دو کلید HMAC ساخته بشه  
 • از این به بعد، تمام پیام های بین کلاینت و سرور رمزنگاری و احراز هویت شده هستن
5. کلاینت HMAC تمام پیام های handshake رو می فرسته
6. سرور هم HMAC تمام پیام های handshake رو می فرسته

- اگه بخوایم یک **TLS connection** رو ببندیم، فرستادن مستقیم **TCP FIN** از طرف Bob به Alice ایده ی خوبی نیست  
 ➤ چون ممکنه باعث **truncation attack** بشه؛ یعنی Trudy وسط جلسه بیاد و اتصال رو زودتر تموم کنه
- Alice ➤ فکر می کنه همه ی داده ها رو گرفته، در حالی که فقط بخشی از داده ها رسیده  
 • راه حل: توی **type field** مشخص کنیم که این رکورد برای پایان دادن به **TLS session** هست

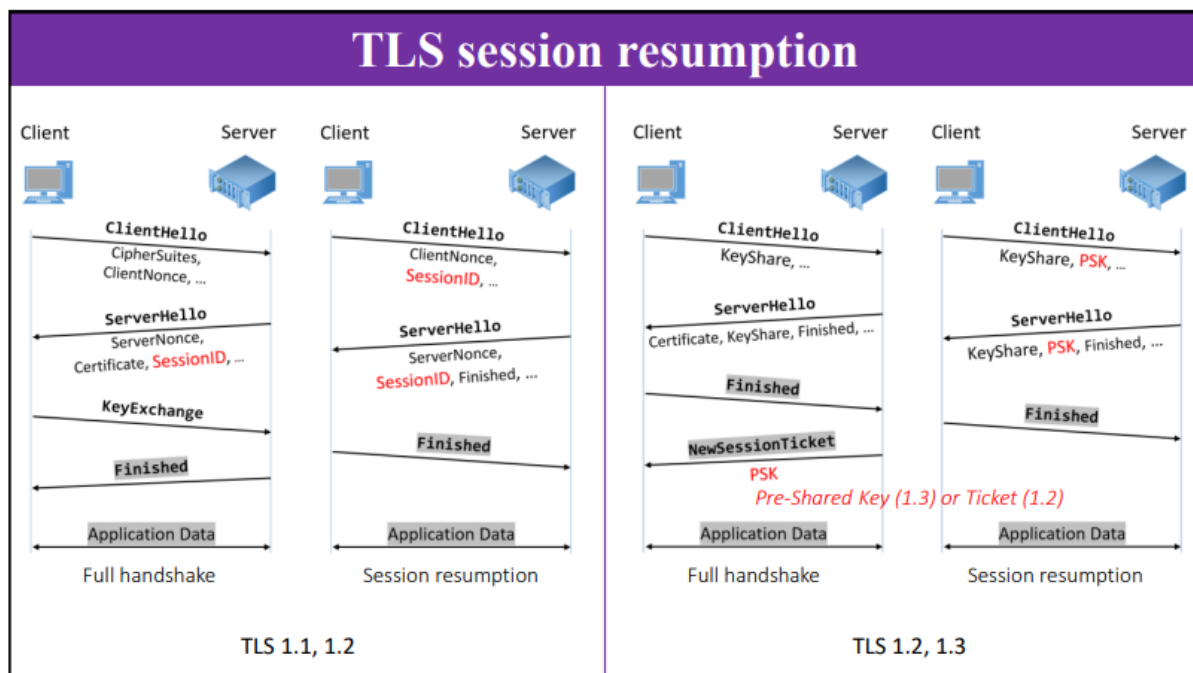
- **Cipher suite** مجموعه ای از الگوریتم های رمزنگاری هست که برای امن کردن ارتباط بین کلاینت و سرور استفاده می شه
- هنگام **TLS handshake**، کلاینت و سرور توافق می کنن کدوم cipher suite رو استفاده کنن
- معمولاً شامل این الگوریتم ها است:
  - Key exchange:** ➤ مثل RSA ، ECDH ، DHE
  - Bulk encryption:** ➤ مثل AES ، DES3
  - Authentication:** ➤ مثل RSA ، ECDSA
  - Message authentication code (HMAC):** ➤ با SHA-256 یا SHA-384

- **Key exchange** (مثل RSA، ECDH، DHE): مشخص می کنه چطوری کلاینت و سرور کلیدهای مشترک برای رمزنگاری رو امن یا هم به اشتراک بذارن.
- **Bulk encryption** (مثل AES، 3DES): خود داده های منتقل شده بین کلاینت و سرور رو رمزنگاری می کنه تا کسی نتونه بخونهشون.
- **Authentication** (مثل RSA، ECDSA): مطمئن می شه که سرور (یا کلاینت) همون کسی هست که ادعا می کنه و جلوی جعل هویت رو می گیره.
- **Message authentication code (HMAC)** (با SHA-256 یا SHA-384): صحت و یکپارچگی داده ها رو چک می کنه تا مطمئن بشیم چیزی در مسیر تغییر نکرده.



### TLS 1.3 Cipher Suites

Cipher Suite Identifier	Algorithms Used
TLS_AES_256_GCM_SHA384	AES with a 256-bit key encryption, 384-bit SHA hash function
TLS_CHACHA20_POLY1305_SHA256	Chacha20–Poly1305 stream cipher, 256-bit SHA hash function
TLS_AES_128_GCM_SHA256	AES with a 128-bit key encryption, 256-bit SHA hash function
TLS_AES_128_CCM_8_SHA256	AES with a 128-bit key encryption, 256-bit SHA hash function
TLS_AES_128_CCM_SHA256	AES with a 128-bit key, SHA-1 hash function



## TLS Session Resumption 🔄

وقتی یک TLS connection بین Alice و Bob برقرار میشه، **Handshake** کامل (یا RSA/DH، گواهی، تولید کلید و غیره) خیلی هزینه‌بره:

- نیاز به چند Round-Trip داره (رفت و برگشت بین کلاینت و سرور)
- از نظر محاسباتی سنگینه (به‌خصوص روی موبایل‌ها)

➡ برای همین، TLS یک روش داره به اسم **Session Resumption** یعنی «ادامه‌دادن جلسه قبلی بدون دوباره انجام دادن کل Handshake».

دو روش معروف داره:

### 1. Session ID 🆔

- موقع اولین Handshake، سرور یه شناسه (Session ID) به کلاینت میده.
  - کلاینت دفعه بعد Session ID رو میفرسته.
  - اگه سرور هنوز اون جلسه رو تو حافظه نگه داشته باشه، می‌تونن با همون کلیدهای قبلی سریع ادامه بدن.
- مشکل: سرور باید اطلاعات همه Session‌ها رو نگه داره (حافظه زیادی می‌خواد).

### 2. Session Ticket 🎫

- برای حل مشکل حافظه، سرور به کلاینت یه **Ticket** میده (یه blob رمزنگاری‌شده شامل کلیدها).
  - کلاینت دفعه بعد Ticket رو می‌فرسته و سرور از توش کلیدها رو بازیابی می‌کنه.
  - نیازی به ذخیره در سمت سرور نیست.
- ✅ این خیلی پرکاربردتره (TLS 1.2 به بعد).

## PSK (Pre-Shared Key) 🔑

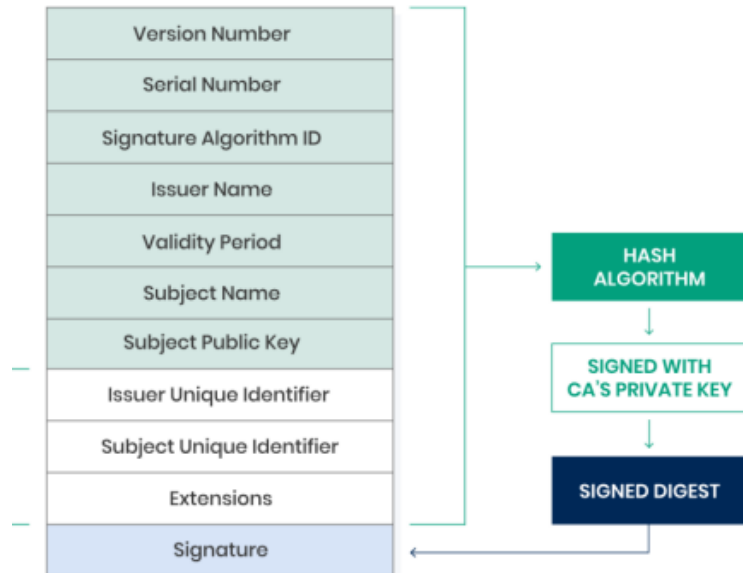
به جای اینکه همیشه کل Handshake با RSA یا DH انجام بشه، میشه از قبل یک **کلید مشترک** بین Alice و Bob وجود داشته باشه.

- **PSK = Pre-Shared Key** یعنی کلید از قبل به اشتراک گذاشته شده.
- توی TLS، PSK می‌تونه جایگزین مرحله‌ی «تولید کلید با RSA/DH» بشه.

مثلاً:

- کلاینت و سرور از قبل یه کلید دارن (یا روش امنی مثل حضوری یا کانال امن دیگه ردوبدل شده).
- توی Handshake فقط به اون کلید اشاره می‌کنن و دیگه نیازی به Public-Key Cryptography (RSA, DH) نیست.

- **X.509** استاندارد ITU که قالب و ساختار گواهی‌های کلید عمومی رو مشخص می‌کنه. این گواهی‌ها در PKI استفاده می‌شن تا ارتباطات و تراکنش‌ها امن بشن.



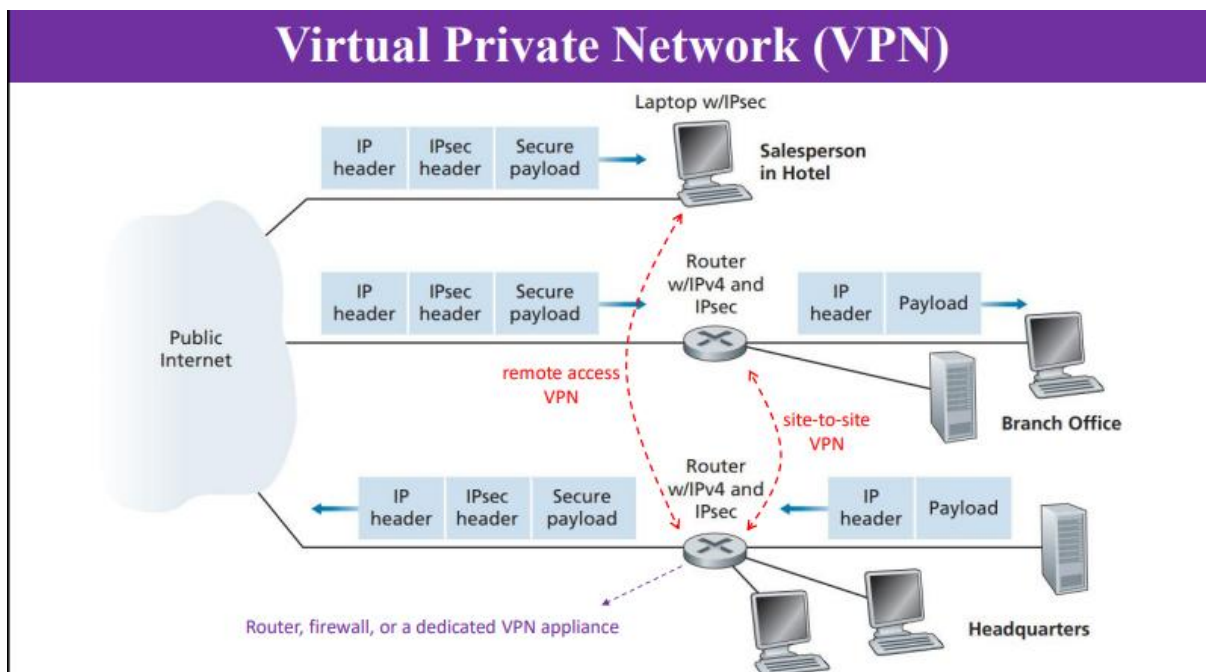
- هر گواهی بخشی از زنجیره اعتماد، از یک **Root CA** معتبر شروع و به **Leaf Certificate** ختم می‌شه.
  - **Root CA Certificate**: گواهی خودامضا که از قبل در مرورگر یا سیستم عامل نصب شده و مورد اعتماد است.
  - **Intermediate CA Certificate(s)**: توسط Root CA صادر می‌شه تا اختیار امضا به یک CA دیگه واگذار بشه.
  - **Leaf Certificate**: گواهی‌ای که در حال بررسیه و توسط Intermediate CA صادر شده.

- وبسایت‌ها برای گرفتن گواهی TLS/SSL باید یک **CSR (Certificate Signing Request)** بسازن و به یک **CA** ارائه بدن.
  - **CSR** شامل نام دامنه، نام سازمان و کلید عمومی است.
- انواع گواهی بر اساس سطح اعتبارسنجی:
  - **Domain Validation (DV)**: مالکیت دامنه رو تایید می‌کنه، سریع و ساده، با اضافه کردن رکورد DNS یا آپلود فایل مشخص.
  - **Organization Validation (OV)**: مالکیت دامنه و اطلاعات سازمان رو تایید می‌کنه.

- Extended Validation (EV): بالاترین سطح اعتماد، بررسی دقیق و کامل، مناسب سایت‌های مالی و تجارت الکترونیک.
- بعد از اعتبارسنجی، CA گواهی SSL/TLS رو صادر می‌کنه.

اسلاید نهم)

- IPsec امنیت رو در لایه شبکه فراهم می‌کنه.
  - بین هر دو موجودیت لایه شبکه (هاست یا روتر) دیتاگرام‌های IP رو امن می‌کنه.
  - برای حفظ محرمانگی، فرستنده محتویات دیتاگرام‌ها (payload) مثل TCP، UDP یا ICMP رو رمزگذاری می‌کنه.
  - علاوه بر رمزگذاری، IPsec قابلیت احراز هویت و یکپارچگی داده رو هم ارائه می‌ده.
- وقتی یه سازمان در مناطق جغرافیایی مختلف فعالیت می‌کنه، می‌خواد شبکه IP خودش رو داشته باشه تا هاست‌ها و سرورهاش بتونن با هم امن و محرمانه ارتباط برقرار کنن.
  - راه حل ۱: ساخت شبکه فیزیکی جداگانه (روتر و لینک‌های اختصاصی) – بهش می‌گن شبکه خصوصی، خیلی گرونه.
  - راه حل ۲: ساخت VPN روی اینترنت عمومی – ترافیک بین دفاتر قبل از ورود به اینترنت رمزگذاری می‌شه تا محرمانگی حفظ بشه.



- **VPN** مثل یه خط ارتباطی امن و اختصاصی عمل می‌کنه، ولی روی شبکه اشتراکی.
- وقتی دو هاست سازمان روی اینترنت عمومی با هم ارتباط برقرار می‌کنن، ترافیک قبل از ورود به اینترنت رمزگذاری می‌شه.
- روتر مرکزی gateway router توی headquarter ها، دیتاگرام IPv4 معمولی رو به **IPsec دیتاگرام** تبدیل می‌کنه.
- دیتاگرام IPsec یه هدر IPv4 سنتی داره تا روترهای اینترنت عمومی مثل یه دیتاگرام معمولی باهاش رفتار کنن.
- وقتی دیتاگرام به لپ‌تاپ می‌رسه، سیستم‌عامل **payload** رو رمزگشایی می‌کنه و داده‌های بدون رمز رو به پروتکل بالاتر (مثل TCP یا UDP) می‌فرسته.

تو مجموعه پروتکل‌های IPsec، دو پروتکل اصلی وجود داره:

1. **Authentication Header (AH)** – فقط هویت فرستنده و صحت داده‌ها رو تضمین می‌کنه، رمزگذاری ندارد.
2. **Encapsulation Security Payload (ESP)** – هویت فرستنده، صحت داده‌ها و رمزگذاری محتوا رو فراهم می‌کنه.

قبل از اینکه دیتاگرام‌های IPsec ارسال بشن، فرستنده و گیرنده یه ارتباط منطقی در لایه شبکه می‌سازن.

- **Security Association (SA):** یک طرفه (simplex) هست و فقط از فرستنده به گیرنده کار می‌کنه.  
 ➤ اگه بخوایم ترافیک امن دوطرفه داشته باشیم، باید دو تا SA درست کنیم، یکی برای هر جهت.
- یه میزبان می‌تونه چندین SA همزمان برای ارتباط با کلاینت‌های مختلف داشته باشه.  
 ➤ همه SA ها در یه پایگاه داده به اسم **Security Association Database (SAD)** ذخیره می‌شن.

همه ترافیک ارسالی از طریق گیت‌وی یا لپ‌تاپ‌ها به اینترنت با IPsec محافظت نمی‌شه.

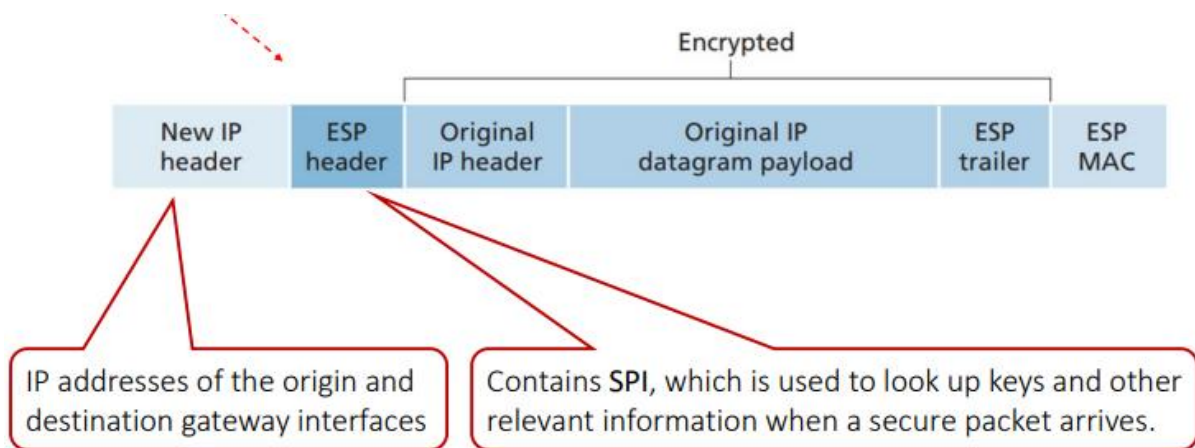
- **Security Policy Database (SPD):** ساختاریه که هر موجودیت IPsec نگه می‌داره.
- SPD مشخص می‌کنه که کدوم نوع دیتاگرام‌ها (با توجه به آدرس مبدا، آدرس مقصد و نوع پروتکل) باید با IPsec پردازش بشن و از کدوم SA استفاده بشه.

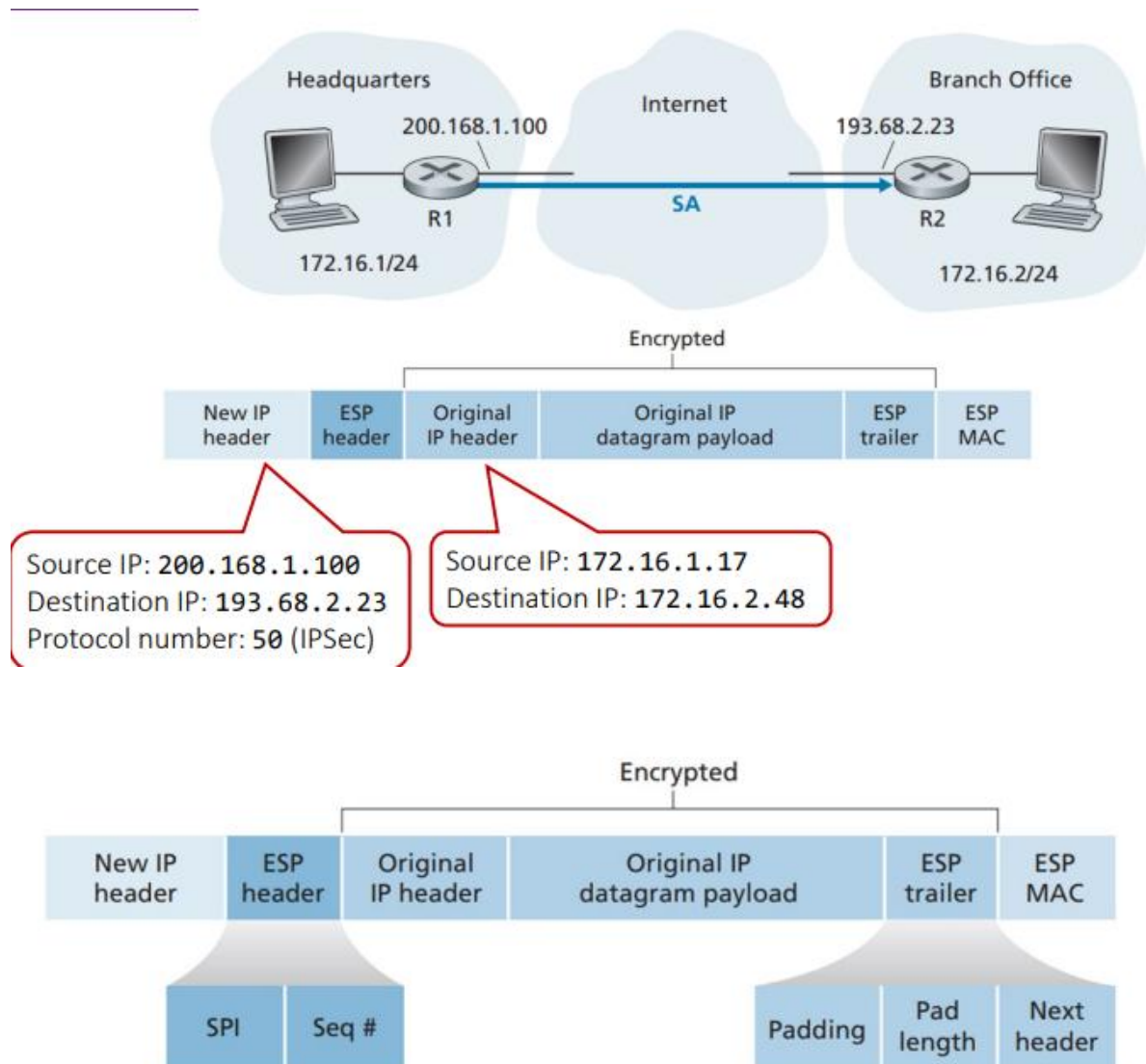
گیتوی روتر اطلاعات وضعیت هر **Security Association (SA)** رو نگه می‌داره:

- **Security Parameter Index (SPI):** شناسه ۳۲ بیتی هر SA
- **Origin & Destination interface:** آدرس‌های مبدا و مقصد SA
- نوع رمزگذاری: مثل 3DES یا CBC
- کلید رمزگذاری و پارامترها مثل Initialization Vector
- نوع بررسی یکپارچگی: مثل HMAC یا MD5
- کلید احراز هویت

IPsec می‌تونه داده‌ها رو در دو حالت مختلف بسته‌بندی کنه:

- **Transport mode** فقط **payload** رمزنگاری میشه (host-to-host).
- **Tunnel mode** کل **IP packet** رمزنگاری میشه و یه **IP header** جدید اضافه میشه. (gateway-to-gateway)





- شماره پروتکل، IP مبدأ و مقصد، و داده‌ی دیتاگرام از دید مهاجم پنهان.
  - مهاجم نمی‌داند بسته شامل چه نوع داده‌ای است. (TCP, UDP, HTTP, ...).
  - محرمانگی IPsec قوی‌تر از SSL است.
  - تغییر بیت‌ها یا جعل روتر با **MAC integrity check** شناسایی می‌شود.
  - حملات Replay با **sequence numbers** قابل تشخیص هستند.
- در VPN با تعداد کم → **end point** تنظیم SA ها (الگوریتم‌ها، کلیدها، SPI به صورت دستی).
  - در شبکه‌های بزرگ و پراکنده → نیاز به مکانیزم خودکار برای ایجاد SA.
  - IPsec این کار رو با پروتکل **IKE (Internet Key Exchange)** انجام می‌دهد.
  - IKE شباهت‌هایی به **handshake در SSL** دارد.

❓ سرور VPN به کلاینت یک آدرس IP خصوصی مجازی اختصاص می‌دهد.

❓ این IP در فرآیند اتصال VPN روی کلاینت پیکربندی می‌شود.

❓ کلاینت بسته‌ها را با همین IP در هدر داخلی ارسال می‌کند.

❓ این IP فقط برای ارتباط با منابع داخلی سازمان (مثل فایل سرور، دیتابیس) استفاده می‌شود.

❓ این آدرس فقط داخل تونل VPN معتبر است و در اینترنت عمومی دیده نمی‌شود.

❓ کلاینت VPN، جدول مسیریابی میزبان را تغییر می‌دهد تا ترافیک خاص از طریق تونل ارسال شود.

**Proxy Server:** ❓ سروری که به عنوان واسطه بین کلاینت و سرور اصلی عمل می‌کند.

❓ در لایه Application کار می‌کند.

❓ نمونه‌ها HTTP Proxy و SOCKS.

**Onion routing:** ❓ تکنیکی که داده‌ها را از طریق چندین میزبان واسطه (relays) می‌فرستد.

❓ داده در سمت کلاینت چندین بار رمزگذاری می‌شود، هر لایه مخصوص یک relay هست.

❓ هر relay فقط لایه خودش را رمزگشایی می‌کند و مقصد بعدی را مشخص می‌کند.

❓ کاربرد: مخفی‌سازی مسیر واقعی و ناشناس‌سازی ارتباطات.

**TOR:** ❓ یک پیاده‌سازی رایگان و متن‌باز از onion routing با شبکه‌ای از relay های داوطلب.



- کنترل ترافیک شبکه: ترافیک ورودی و خروجی باید بررسی امنیتی، ثبت، رد یا هدایت شود.
- دستگاه‌های عملیاتی: فایروال‌ها، سیستم‌های تشخیص نفوذ (IDS) و سیستم‌های پیشگیری از نفوذ (IPS) این کار را انجام میدن.
- هدف: بخشی از استراتژی دفاع چندلایه‌ای در شبکه.

فایروال شبکه: دستگاه امنیتی که ترافیک ورودی و خروجی را بر اساس قوانین امنیتی مانیتور و فیلتر می‌کند.

وظایف:

- مسدود کردن دسترسی‌های غیرمجاز
- اجازه دادن به ترافیک قانونی

مکان نصب: معمولاً در مرز شبکه قرار می‌گیرد.

- فایروال‌های تجاری معمولاً به صورت سخت‌افزارهای تخصصی عرضه می‌شوند.
- می‌توان یک فایروال مبتنی بر packet filter را با لینوکس و iptables ساخت.
- امروزه فایروال‌ها اغلب در روترها پیاده‌سازی شده و با استفاده از SDN به صورت از راه دور کنترل می‌شوند.

- دسته‌بندی فایروال‌ها:

1. Traditional packet filters
2. Stateful filters
3. Application gateways

- Packet filter: بررسی جداگانه هر دیتاگرام و تصمیم برای عبور یا مسدود کردن آن بر اساس قوانین مدیر شبکه
- معیارهای فیلتر کردن:
  - آدرس مبدا و مقصد IP
  - نوع پروتکل در فیلد (TCP, UDP, ICMP, OSPF) و غیره
  - پورت‌های مبدا و مقصد TCP/UDP

- بیت‌های فلگ ( TCP مثل SYN ، ACK)
- نوع پیام ICMP
- قوانین متفاوت برای دیتاگرام‌های ورودی و خروجی
- قوانین متفاوت برای اینترفیس‌های مختلف روتر

## Policies

- A network administrator configures the firewall based on the policy of the organization.

Policy	Policy
No outside Web access.	Drop all outgoing packets to any IP address, port 80.
No incoming TCP connections, except those for organization's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80.
Prevent video streaming from eating up the available bandwidth.	Drop all incoming UDP packets—except DNS packets.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP ping packets going to a "broadcast" address (eg 130.207.255.255).
Prevent your network from being tracerouted.	Drop all outgoing ICMP TTL expired traffic.

- فیلتر کردن می‌تواند بر اساس بیت ACK در TCP باشد
- مثال:
  - اجازه دادن به کلاینت‌های داخلی برای اتصال به سرورهای خارجی
  - جلوگیری از اتصال کلاینت‌های خارجی به سرورهای داخلی
  - راه حل: مسدود کردن تمام بسته‌های ورودی که بیت ACK آنها برابر 0 است

- قوانین فایروال با **Access Control List (ACL)** در روترها پیاده‌سازی می‌شوند
- هر اینترفیس روتر لیست قوانین مخصوص خودش را دارد
- قوانین به ترتیب از بالا به پایین روی هر دیتاگرامی که از اینترفیس عبور می‌کند اعمال می‌شوند

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	—
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	—
deny	all	all	all	all	all	all

- **Stateful filters** وضعیت (state) اتصالات TCP را دنبال می کنند
- از این اطلاعات برای گرفتن تصمیم های هوشمندانه درباره ی اجازه عبور یا مسدود کردن بسته ها استفاده می کنند

## Stateful packet filters

- Connection table for a stateful packet filter (firewall)

		source address	dest address	source port	dest port
		222.22.1.7	37.96.87.123	12699	80
		222.22.93.2	199.1.205.23	37654	80
		222.22.65.143	203.77.240.43	48712	80

action	source address	dest address	protocol	source port	dest port	flag bit	check connection
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any	
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK	X
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	—	
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	—	X
deny	all	all	all	all	all	all	

- **Application gateways** داده های اپلیکیشن را بررسی می کنند و تصمیم امنیتی می گیرند، نه فقط هدر شبکه
- همه ی ترافیک ورودی و خروجی اپلیکیشن از این سرور عبور می کند
- امکان تشخیص نوع ترافیک (مثلاً وب گردی HTTP از فایل شیرینگ P2P) وجود دارد
- مثال **Web Application Firewall (WAF)** :

- Web Application Firewalls (WAF) برای شناسایی و مسدود کردن ترافیک مخرب که وب اپلیکیشن‌ها را هدف قرار می‌دهد استفاده می‌شوند.
- WAF ترافیک HTTP/HTTPS بین کلاینت (مثل مرورگر کاربر) و وب اپلیکیشن را بررسی می‌کند.
- WAF به دنبال الگوها یا رفتارهایی است که با تکنیک‌های حمله شناخته شده یا فعالیت‌های غیرعادی مطابقت دارند.

#### WAF deployment models:

- **Transparent:** ترافیک HTTP مستقیم به وب اپلیکیشن می‌رود و WAF بین کلاینت و سرور قرار می‌گیرد بدون نیاز به تغییر DNS یا پیکربندی اپلیکیشن.
- **Cloud-Based:** ارائه دهنده ابری WAF را مدیریت می‌کند و ترافیک از طریق شبکه‌ی آن‌ها برای فیلتر شدن عبور می‌کند.
- **Proxy-Based (Reverse Proxy):** ترافیک کلاینت مستقیم به WAF می‌رود و WAF ترافیک فیلتر شده را به وب اپلیکیشن ارسال می‌کند.

#### WAF detection methods:

- **Signature-based detection:**
  - از دیتابیس از پیش تعریف شده‌ی امضاها برای حمله استفاده می‌کند.
  - امضاها رشته‌ها، عبارات منظم یا توالی‌هایی هستند که با حملات شناخته شده مطابقت دارند.
  - مثال‌ها:
    - درخواست‌های SQL با الگوهای مشکوک مثل `SELECT * FROM users` OR `'1'='1` یا `WHERE username = 'admin'`
    - اسکریپت‌های `<script>` که می‌توانند حمله XSS باشند.

- **Behavioral analysis:**
  - بررسی الگوهای ترافیک و شناسایی انحراف‌ها از رفتار معمول.
  - درخواست‌هایی که الگوهای غیرعادی دارند را علامت گذاری یا مسدود می‌کند.
  - مثال‌ها:
    - تلاش‌های مکرر و سریع برای ورود به سیستم → بلاک کردن IP.
    - ورود از موقعیت جغرافیایی جدید یا با هدرهای غیرمعمول → هشدار.
    - کاهش اثر حملات DDoS.

- **Rule/policy-based filtering:**

- تعریف قوانین یا سیاست‌های سفارشی برای مسدود یا اجازه دادن به انواع خاص ترافیک.
- قوانین می‌توانند بر اساس آدرس IP ، هدرهای درخواست ، URL یا محتوای payload باشند.
- مثال‌ها:
  - مسدود کردن دسترسی به URL های حساس مثل /admin یا /config
  - مسدود کردن درخواست‌هایی با کلمات کلیدی خاص مثل DROP TABLE یا eval().

- **AI-based detection:**

- استفاده از هوش مصنوعی برای شناسایی الگوهای غیرمعمول و حملات ناشناخته.

- **Reputation-Based Detection:**

- بررسی شهرت IP یا منبع ترافیک و مسدود کردن منابع مخرب شناخته شده.

- **Application gateways واقعاً Deep Packet Inspection (DPI) انجام می‌دهند.**

- **DPI:** بررسی فراتر از هدرها و نگاه کردن به داده‌های واقعی اپلیکیشن که بسته‌ها حمل می‌کنند.

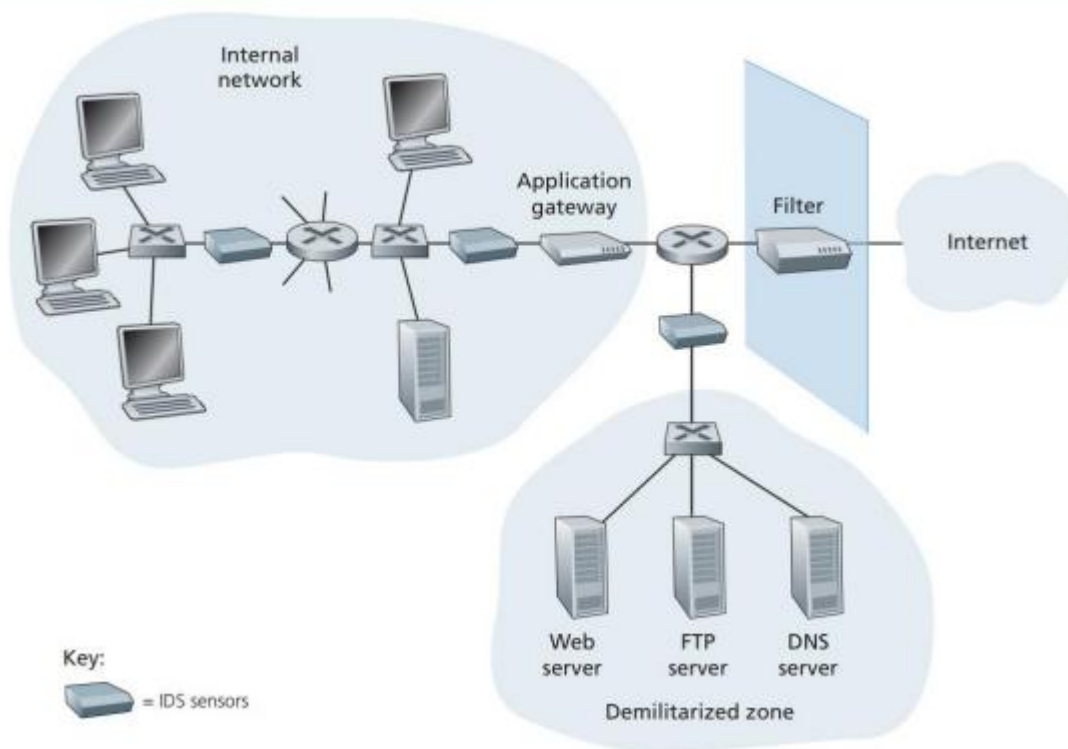
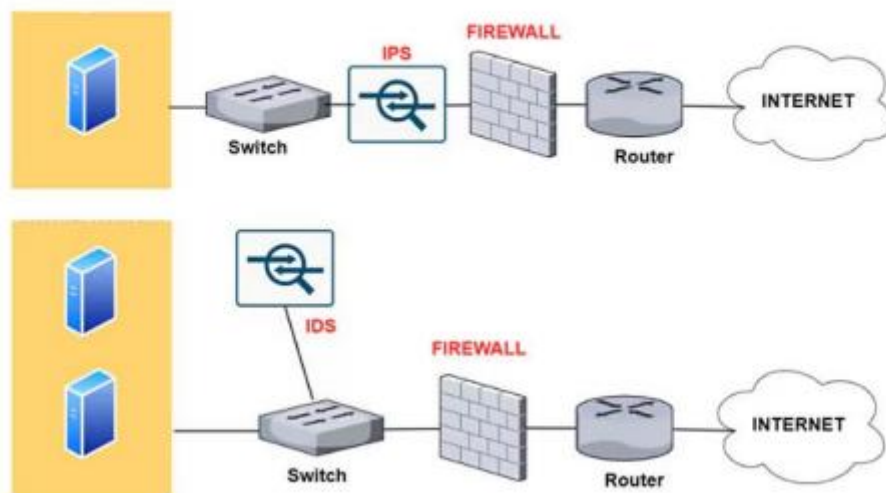
- **Application gateway فقط برای یک اپلیکیشن خاص DPI انجام می‌دهد.**

- **Intrusion Detection System (IDS):** تمام بسته‌ها را بررسی می‌کند و DPI انجام می‌دهد.

- اگر بسته یا مجموعه‌ای از بسته‌ها مشکوک باشد، هشدار می‌دهد.
- IDS می‌تواند حملات مختلف را شناسایی کند:

- Network mapping
- Port scans
- TCP stack scans
- DoS/flooding attacks
- Worms و ویروس‌ها
- حملات ضعف OS و اپلیکیشن‌ها

- دستگاهی که ترافیک مشکوک را فیلتر می‌کند، **Intrusion Prevention System (IPS)** نام دارد.



• IDS دو نوع اصلی دارد Signature-based و Anomaly-based.

• Signature-based IDS:

- یک بانک اطلاعاتی از امضاهای حمله دارد.
- هر امضا شامل مجموعه‌ای از قوانین مرتبط با فعالیت نفوذ است.
- امضا می‌تواند مربوط به یک بسته باشد (مثلاً هدر یا رشته خاصی در payload) یا به یک سری بسته‌ها مرتبط باشد.
- نمی‌تواند حملات جدید را تشخیص دهد.

- **Anomaly-based IDS:**

- یک پروفایل ترافیک طبیعی ایجاد می کند.
- سپس دنبال جریان هایی می گردد که از نظر آماری غیرعادی هستند.

- بسیاری از سیستم های IDS اختصاصی هستند.

- **Snort** یک IDS/IPS متن باز و رایگان است:

- قابل اجرا روی Linux، UNIX و Windows.
- از **libpcap** برای شنود بسته ها استفاده می کند.
- جامعه بزرگی از کاربران و متخصصان امنیت دارد که بانک امضا های آن را به روز نگه می دارند.
- معمولاً چند ساعت بعد از حمله جدید، یک امضا منتشر می شود و صدها هزار نصب Snort در سراسر جهان آن را دریافت می کنند.