الجامعة الافتراضية السورية

SYRIAN VIRTUAL UNIVERSITY

مقدمة في التشفير

الدكتور صلاح الدوه جي

Books

مقدمة في التشفير

الدكتور صلاح الدوه جي

من منشورات الجامعة الافتراضية السورية

الجمهورية العربية السورية 2018

Cryptography

Salah DOWAJI

Publications of the Syrian Virtual University (SVU)

Syrian Arab Republic, 2018

الفهرس

مقدمة عامة

1. مقدمة

ساهمت شبكة الانترنت بتغيير حياتنا اليومية بعدة طرق، وذلك على اعتبار أن هذه الشبكة أضحت وسيلة الاتصال العالمية:

فقد سمحت هذه الشبكة للأفراد والمؤسسات حول العالم بالمشاركة بالمعلومات وبالتواصل التجاري وتبادل رسائل البريد الالكتروني. مما جعل الاعتماد على شبكة الانترنت أمراً ملحاً ولا يمكن تفاديه.

وعلى اعتبار أن الانترنت تشبه في تكوينها بنية المنتدى المفتوح والمتاح للجميع، أدى ذلك إلى ظهور الكثير من المشاكل الأمنية، حيث برزت الحاجة لتوفير:

- السرية **Confidentiality**.
- تكامل المعطيات **Integrity**.
- تحديد هوية المتصل **Authentication**.

يعد التشفير **Cryptography** أحد أهم الوسائل المستخدمة لتوفير بيئة آمنة لتبادل المعلومات وحمايتها.

نعرض فيما يلي أسس ومبادئ تشفير المعلومات ونبين بشكل سريع كيفية استخدام هذه التقنيات لتحقيق الأهداف الثلاث التي تحدثنا عنها فيما سبق.

يهدف هذا المقطع إلى توضيح الأفكار التالية:

1. تعريف الأهداف الثلاثة لأمن المعلومات.
2. تعريف الهجمات **Attacks** التي تهدد الأهداف الثلاثة لأمن المعلومات.
3. تعريف الخدمات الأمنية **Security Services** وتوضيح علاقتها كيف بالأهداف الثلاثة لأمن المعلومات.
4. تحديد الأدوات الأمنية **Security Mechanisms** المستخدمة لتوفير الخدمات الأمنية.
5. عرض تقنيتي: التشفير **Cryptography** والتورية **Steganography** المستخدمة لبناء وتحقيق الأدوات الأمنية.

حتى عقود سالفة كان يجري تخزين المعلومات ضمن ملفات ومصنفات ورقية. يجري ضمان سرية هذه الملفات من خلال تخزينها ضمن أماكن محمية ولا يمكن الوصول إليها أو تعديلها من دون تصريح خاص أو من قبل عدد محدود ومحدد من الأشخاص.

كان يجري توفير إتاحية الوصول إلى هذه المعلومات من خلال تعيين شخص واحد على الأقل قادر على الوصول إلى هذه المعلومات في أي وقت.

مع ظهور الحواسيب، بدأ التفكير بتخزين المعلومات إلكترونياً بدلاً من تخزينها ورقياً. ولكن بقيت متطلبات أمنها الثلاث السابقة هي نفسها:

السرية والتكامل والإتاحية.

أما المختلف في هذه الوضعية الجديدة فهو طريقة تحقيق هذه المتطلبات.

ساعد استخدام الحواسيب خلال فترة العقدين السابقين على خلق ثورة في طريقة التعامل مع المعلومات واستخدامها. المعلومات في زمننا الحالي موزعة ومخزنة في عدة أماكن ويمكن للأشخاص المصرح لهم بالوصول إليها سواء كان محلياً أو عن بعد من خلال الشبكة الحاسوبية.

أدى ذلك كله أن تأخذ متطلبات أن المعلومات الثلاثة التي تحدثنا عنها فيما سبق أبعاداً أخرى، إذ لا يكفي أن يجري المحافظة على سرية المعلومات أثناء تخزينها وإنما يجب تحقيق ذلك أثناء نقلها وإرسالها من حاسوب إلى آخر.

سنتحدث فيما يلي عن الأهداف الرئيسية الثلاثة لأمن المعلومات. ثم نبين كيف يمكن أن تحدث الهجمات التي تشكل تهديداً للأهداف أمن المعلومات.

نعرض بعد ذلك الخدمات الأمنية المرتبطة بأهداف أمن المعلومات ونبين بعدها الأدوات الأمنية الواجب الحصول عليها لتوفير الخدمات الأمنية. نعرض أخيراً التقنيات التي يمكن استخدامها لبناء الأدوات الأمنية.

2. أهداف أمن المعلومات Security Goals

نقوم في هذه الفقرة بتعريف أهداف أمن المعلومات التي تحدثنا عنها في الفقرة السابقة.

- **السرية** Confidentiality: يعد ذلك من أهم أهداف أمن المعلومات. نحن كأشخاص أو مؤسسات بحاجة إلى حماية معلوماتنا السرية لعدة أسباب لها علاقة بعملنا وبنواحي عملية مختلفة. لا يطبق هذا الهدف على المعلومات عند تخزينها فقط ولكن يطبق أيضاً أثناء نقلها وتبادلها.
- **التكامل** Integrity: نحتاج إلى تبادل المعلومات من وقت إلى آخر. يعني تكامل المعطيات

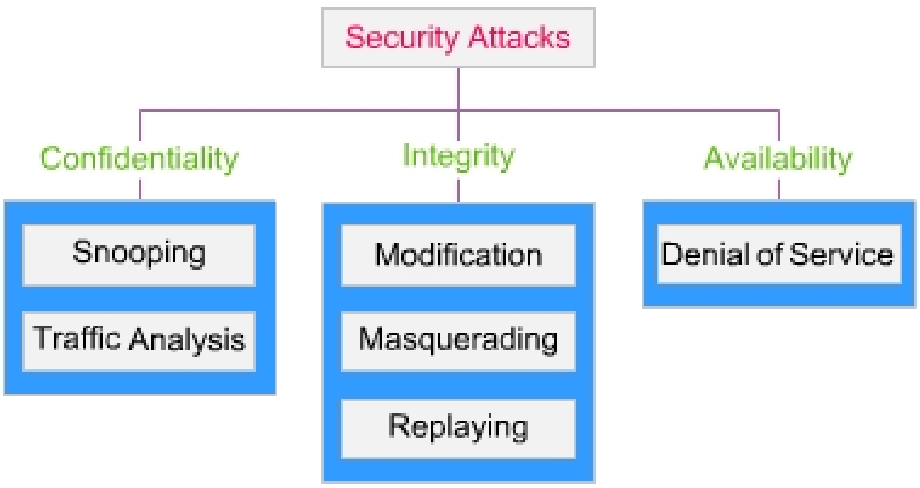
أن:

تعديلها يجب أن يتم حصرا من قبل الأشخاص المصرح لهم بذلك. يجب ملاحظة أن تعديل المعلومات لا يمكن أن يحدث بنتيجة فعل غير قانوني فقط، وإنما يمكن أن يؤدي انقطاع التيار الكهربائي في بعض الحالات إلى تعديل بعض المعلومات.

- **الإتاحية Availability**: يجب أن تتوفر إمكانيات إتاحة المعطيات للجهات المصرح لها بذلك حصراً، حيث تعتبر المعلومات غير المتاحة لأحد غير مفيدة لأحد. كما تحتاج المعلومات إلى أن يجري تعديلها من وقت إلى آخر ولكن يجب أن يجري ذلك من قبل من يملك تصريح بذلك. يمكن أن يؤدي عدم توفر الإتاحية إلى نتائج سيئة مماثلة لتلك الناتجة عن عدم مراعاة سرية المعلومات.

3. الهجمات Attacks

يمكن تهديد أهداف أمن المعلومات التي تحدثنا عنها في الفقرة السابقة بواسطة عدد من الهجمات. ورد في مراجع مجال أمن المعلومات عدة تصنيفات لهذه الهجمات التي سنقوم فيما يلي بفرزها ضمن ثلاث مجموعات حسب هدف أمن المعلومات الذي تهدده، كما هو مبين في الشكل التالي:



وسنشرح فيما يلي معنى هذه الهجمات المختلفة.

الهجمات التي تهدد السرية Attacks Threatening Confidentiality

يتوفر بشكل عام نوعين من الهجمات التي تشكل تهديداً للسرية، هما:

- **التصيد** Snooping: ويرمز ذلك للوصول غير المصرح به للمعطيات أو التقاط معطيات سرية أثناء نقلها على شبكة الانترنت، ثم استخدام هذه المعطيات لأغراض مسيئة. لمنع التصيد يمكن جعل المعطيات غير مفيدة للأطراف الأخرى باستخدام أحد وسائل التشفير التي سنتحدث عنها لاحقاً.
- **تحليل المعطيات المتبادلة** Traffic analysis: رغم استخدام وسائل التشفير لجعل المعطيات غير مفيدة للأطراف الأخرى، إلا أنه يمكن الحصول على معلومات أخرى مفيدة من خلال تحليل ومراقبة المعلومات المتبادلة من خلال الشبكة. يمكن مثلاً: معرفة العنوان الإلكتروني للمرسل أو المستقبل، أو يمكن تجميع الطلبات والردود عليها لمحاولة معرفة طبيعة حركة المعطيات المنفذة.

الهجمات التي تهدد التكامل Attacks Threatening Integrity

يمكن تهديد تكامل المعطيات بعدة أنواع من الهجمات، التي نذكر منها:

- **التغيير** Modification: بعد تصيد المعطيات المتبادلة أو الوصول إليها، يمكن للمهاجم تغييرها لجعلها مفيدة له. كما يمكن أن يقوم المهاجم بحذف أو تأخير وصول رسالة لفائدة مرتبطة به أو لتعطيل النظام.
- **انتحال الصفة** Masquerading: ويحدث ذلك عندما يقوم المهاجم بانتحال صفة شخص آخر. كما يمكن أن يتظاهر المهاجم بكونه الطرف المستقبل للرسائل.
- **الإعادة** Replaying: يقوم المهاجم بإعادة الاتصال مع المستقبل باستخدام نفس الرسائل التي أرسلها المرسل بعد قيامه بنسخها أثناء عملية اتصال حقيقية بين الطرفين.
- **الإنكار** Repudiation: يختلف هذا الهجوم عن غيره لأنه يحتاج إلى طرف أو طرفي الاتصال لإنجازه المرسل أو المستقبل. يمكن لمرسل الرسالة أن ينكر لاحقاً أنه أرسل الرسالة، أو يمكن للمستقبل أن ينكر لاحقاً استقباله للرسالة.

الهجمات التي تهدد الإتاحية Attacks Threatening Availability

سنتحدث هنا عن نوع واحد من الهجمات التي تهدد الإتاحية، ألا وهو:

تعطل الخدمة Denial of service (DoS):

وهو نوع شائع من الهجمات التي تؤدي إلى إبطاء النظام الحاسوبي المستهدف أو قد تؤدي إلى توقفه تماماً. يمكن أن يتبع المهاجم عدة استراتيجيات لتنفيذ ذلك.

يمكن مثلاً أن يقوم بإرسال كمية كبيرة من الطلبات مما يؤدي إلى تعطل النظام الحاسوبي المستهدف أو يمكن أن يعترض المهاجم أجوبة المخدم بتعديلها أو حذفها مما يظهر المخدم كما لو أنه لا يتجاوب مع طلبات الزبائن.

يمكن أيضاً تصنيف الهجمات بطريقة أخرى ضمن فئتين حسب طريقة عملها، كما هو مبين في الجدول التالي:

الهجمات	الفئة (فعال أو غير فعال)	التهديد الذي يمثله
التصيد Snooping المعطيات المتبادلة Traffic analysis	غير فعال	السرية Confidentiality
التغيير Modification انتحال الصفة Masquerading الإعادة Replaying الإنكار Repudiation	فعال	التكامل Integrity
تعطل الخدمة Denial of service	فعال	الإتاحية Availability

الهجمات غير الفعالة Passive Attacks:

يكون هدف المهاجم في هذا النوع من الهجمات تجميع المعلومات، فهو لا يقوم بتعديل المعلومات أو بالتسبب بأي أذى للنظام الحاسوبي. يمكن لعملية تجميع المعلومات أن تسبب أذى للمرسل أو المستقبل لكنها لا تؤثر على النظام الحاسوبي الذي يتابع عمله بشكل طبيعي. تعتبر الهجومات المهددة للسرية هجمات غير فعالة، وهي هجمات من الصعب جداً التقاطها حتى يكتشف المرسل أو المستقبل تسرب معلومات سرية تخصه. يمكن منع هذا النوع من الهجومات من خلال تشفير المعطيات.

الهجومات الفعالة Active Attacks:

يتسبب هذا النوع من الهجومات بتغيير المعطيات أو بأذى للنظام الحاسوبي. تعتبر الهجومات المهددة للتكامل والإتاحية من هذا النوع، ومن السهل التقاطها أكثر من منعها لأن المهاجم يقوم بتنفيذها بعدة أساليب وطرق.

4. الخدمات والأدوات Services and Mechanisms

يقدم اتحاد الاتصالات العالمي - قسم معايرة الاتصالات **International Telecommunication Union -Telecommunication Standardization Sector (ITU-T)** بعض من الخدمات الأمنية ويوفر عدد من الأدوات لبنائها. لا يوجد فرق كبير بين الخدمات والأدوات لأنه في بعض الحالات يمكن لأداة ولتجميع عدد منها أن يشكل خدمة. كما يمكن استخدام نفس الأداة مع عدة خدمات. نقوم فيما يلي بإعطاء فكرة عامة عن:

1. الخدمات الأمنية.
2. الأدوات الأمنية المهمة.

الخدمات الأمنية Security Services

عرف المعيار **(X.800) ITU-T** خمس خدمات مرتبطة بالأهداف الأمنية والهجمات التي تحدثنا عنها في الفقرات السابقة. نعرض فيما يلي بشكل مختصر فكرة عن هذه الخدمات التي من السهل الربط بينها وبين هدف أو أكثر من أهداف أمن المعلومات. من السهل أيضاً أن نلاحظ أن هذه الخدمات من أجل منع الهجومات التي تحدثنا عنها فيما سبق.

- **سرية المعطيات Data Confidentiality**: وهي خدمة مصممة من أجل حماية المعطيات ضد الهجومات التي تستهدف الكشف عنها. تتضمن هذه الخدمة ضمان سرية الرسائل بكاملها أو جزء منها والحماية ضد هجمات تحليل المعطيات والتصيد.
- **تكامل المعطيات Data Integrity**: وهي خدمة مصممة لحماية المعطيات ضد التغييـر أو الإضافة أو الحذف أو إعادة التنفيذ. يمكن لها حماية كامل الرسالة أو جزء منها.
- **التعرف / تحديد الهوية Authentication**: وتفيد في التعرف وتحديد هوية الطرف الآخر المتصل **(Peer entity authentication)**. يفيد ذلك في تحديد هوية المرسل أو المستقبل في حال الاتصالات الموجهة بالربط **Connection-oriented communication**. كما أنها تفيد في تحديد مصدر المعطيات **(Data origin authentication)** في حالة الاتصالات غير الموجهة بالربط **Connectionless communication**.
- **منع الإنكار Nonrepudiation**: تفيد هذه الخدمة بالحماية ضد إنكار المرسل أو المستقبل بتنفيذ عملية الاتصال. يستطيع المستقبل مع خدمة منع الإنكار المزودة بوسيلة الدلالة على المصدر **Proof of origin** أن يحدد مصدر المعلومات وهوية المرسل. أما مع خدمة منع الإنكار المزودة بوسيلة الدلالة على الاستلام **Proof of delivery** فيمكن للمرسل أن يحدد

مكان استلام المعطيات وهوية المستقبل.

- **التحكم بالوصول** Access Control: توفر هذه الخدمة الحماية ضد الوصول غير المصرح به إلى المعطيات. وتتضمن كلمة الوصول إلى المعطيات كل ما له علاقة بعمليات القراءة والكتابة والتعديل والتنفيذ للبرامج وغيرها من العمليات من هذا النمط.

الأدوات الأمنية Security Mechanisms

عرف المعيار (X.800) ITU-T أيضاً عدداً من الأدوات الأمنية اللازمة لتوفير الخدمات الأمنية التي تحدثنا عنها في الفقرة السابقة. نعرض فيما يلي بشكل مختصر فكرة عن هذه الأدوات:

- **التشفير** Encipherment: تفيد هذه الأداة في إخفاء المعطيات للحفاظ على سريتها. يمكن أيضاً استخدامها كتتمة وتكملة للأدوات الأخرى اللازمة لبناء خدمة أمنية. يتوفر لدينا حالياً تقنيتان للتشفير، هما:

  التعمية Cryptography والتورية Steganography اللتين سنتحدث عنهما في الفقرة التالية.

- **تكامل المعطيات** Data Integrity: تفيد هذه الأداة في إضافة بعض المعلومات الإضافية (مجاميع قصيرة للتحقق Short checkvalues) وفقاً لإجراء معين إلى المعطيات نفسها قبل إرسالها. يقوم المستقبل باستقبال المعطيات مع مجاميع التحقق، ثم يحاول حساب هذه المجاميع بنفسه على المعطيات الحقيقية للتحقق من صحة المجاميع. إذا تطابقت المجاميع فهذا يعني أنه جرى المحافظة على تكامل المعطيات.
- **التوقيع الإلكتروني** Digital Signature: تفيد هذه الأداة في توفير إمكانية توقيع المرسل للمعطيات المرسلة من قبله الكترونياً بحيث يمكن للمستقبل أن يتحقق من هذا التوقيع عند استلام المعطيات للتأكد من هوية المرسل. يتم ذلك باستخدام المفتاح الخاص Private Key بالمرسل لتوقيع المعطيات قبل إرسالها، ثم يستخدم المستقبل المفتاح العام Public Key للمرسل والمعروف من قبل الجميع للتحقق من صحة التوقيع على نفس المعطيات المستقبلة من قبله.
- **تبادل التعارف** Authentication Exchange: تعني هذه الأداة أن طرفي الاتصال يتبادلان بعض الرسائل ليتحقق كلاً منهما من هوية الآخر. يمكن مثلاً أن:

  يثبت أحد الطرفين أنه يعلم سراً لا يفترض بأحد آخر أن يعرفه.

- **حشو المعطيات** Traffic Padding: ويفيد ذلك في إدراج كمية من المعطيات غير المفيدة

ضمن المعطيات الحقيقية بحيث تصبح هذه المعطيات بمجملها مضللة لهجمات تحليل حركة المعطيات.

- التحكم بالتوجيه **Routing Control**: ويعني ذلك اختيار طرق توجيه مختلفة بين المرسل والمستقبل والتبديل بينها من وقت إلى آخر لتضليل المتصنت على أحد هذه الطرق المتاحة.
- **المرور بطرف ثالث** **Notarization**: ويعني ذلك اختيار طرف ثالث موثوق به للتحكم بالاتصال بين طرفين آخرين. يمكن اللجوء إلى هذه الأداة لمنع الإنكار. يمكن مثلاً أن: يلجأ المستقبل باختيار طرف ثالث موثوق به لتخزين طلبات المرسل بهدف منعه من الإنكار لاحقاً بأنه أرسل هذه الطلبات.
- **التحكم بالوصول** **Access Control**: يمكن اللجوء إلى هذه الأداة لتحديد سماحيات وصول المستخدم إلى المعطيات والمصادر المملوكة من قبل النظام، مثل: السماحيات الممنوحة وفقاً لكلمات السر **Passwords** أو الرموز السرية **PINs**.

العلاقة بين الخدمات والأدوات Relation between Services and Mechanisms

يوضح الجدول التالي العلاقة بين الخدمات الأمنية والأدوات الأمنية التي سبق لنا الحديث عنها:

الخدمة الأمنية	الأداة / الأدوات الأمنية
سرية المعطيات **Data Confidentiality**	التشفير **Encipherment** والتحكم بالتوجيه **Routing Control**.
تكامل المعطيات **Data Integrity**	التشفير **Encipherment** والتوقيع الإلكتروني **Digital Signature** وتكامل المعطيات **Data Integrity**.
التعرف / تحديد الهوية **Authentication**	التشفير **Encipherment** والتوقيع الإلكتروني **Digital Signature** وتبادل التعارف **Authentication Exchange**.
الإنكار **Nonrepudiation**	التوقيع الإلكتروني **Digital Signature** وتكامل المعطيات **Data Integrity** والمرور بطرف ثالث **Notarization**.
التحكم بالوصول **Access Control**	أداة التحكم بالوصول **Access Control**

5. التقنيات Techniques

تعتبر التقنيات التي نتحدث عنها هنا هي تقنيات نظرية من أجل تحقيق أمن المعلومات، حيث يتطلب تحقيق الأهداف الأمنية استخدام بعض التقنيات. من هذه التقنيات لدينا اليوم:

1. التعمية **Cryptography**.
2. التورية **Steganography**.

التعمية Cryptography

يمكن تحقيق العديد من الأدوات الأمنية التي تحدثنا عنها سابقاً باستخدام تقنيات التعمية **Cryptography** التي تعني باليونانية:

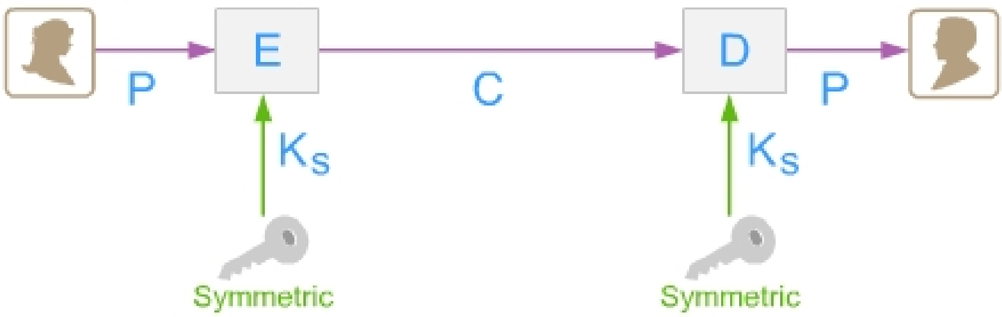
كتابة الأسرار أو الكتابة السرية **Secret writing**.

لكن هذه الكلمة ترمز بالنسبة لنا إلى علم وفن تحويل الرسائل لتصبح آمنة ضد كافة أشكال الهجوم. كانت هذه التقنية تقتصر في الماضي على أساليب التشفير **Encryption** وفك التشفير **Decryption** للرسائل المتبادلة باستخدام مفاتيح سرية، أما اليوم فهي تدل على ثلاث تقنيات مختلفة، وهي:

1. التشفير بمفاتيح متناظرة **Symmetric-key encipherment**.
2. التشفير بمفاتيح غير متناظرة **Asymmetric-key encipherment**.
3. التهشير **Hashing**.

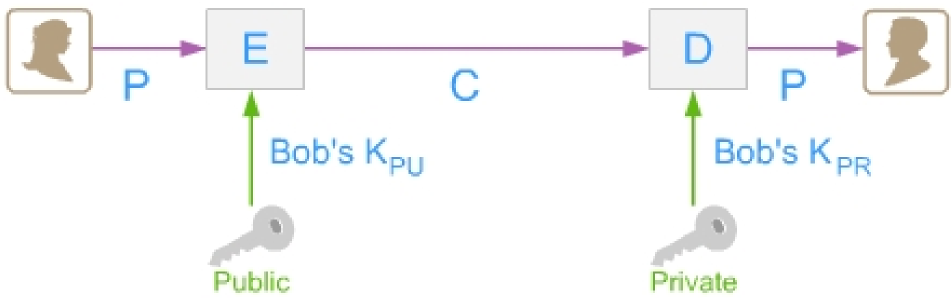
التشفير بمفاتيح متناظرة Symmetric-key encipherment

يجري التشفير وفقاً لهذه التقنية باستخدام مفتاح سري **Secret-key** مشترك من قبل الطارفين المتصلين كما هو مبين في الشكل التالي:



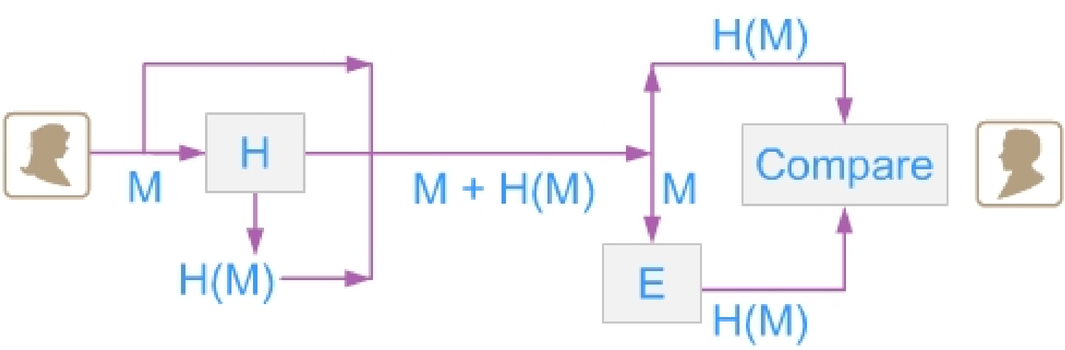
التشفير بمفاتيح غير متناظرة Asymmetric-key encipherment

يجري التشفير وفقاً لهذه التقنية باستخدام مفتاحين أحدهما معلن وهو المفتاح العام **Public Key** والآخر سري وهو **Private key**. ولإرسال رسالة مشفرة من طرف إلى آخر يجب تشفير معطياتها باستخدام المفتاح العام المعلن للطرف المستقبل الذي يقوم بدوره فك تشفير الرسالة بعد وصلها بواسطة مفتاح الخاص السري، ولن يتمكن أي شخص آخر من تنفيذ ذلك لأنها شفرت في الأصل بمفتاحه العام المرتبط به، كما هو مبين في الشكل التالي:



التهشير Hashing

يجري وفقاً لهذه التقنية إنشاء تلخيص من طول ثابت لمحتوى الرسالة **Fixed-length message digest** وهو أصغر بكثير من الرسالة. يجري بعدها إرسال الرسالة وملخصها إلى الطرف الثاني الذي يعيد بدوره إنشاء ملخص للرسالة المستقبلة لمطابقته مع ملخص الرسالة المرسلة في الأصل. إذا حدث التطابق فهذا يعني أن تكامل المعطيات محقق، كما هو مبين في الشكل التالي:



التورية Steganography

تعني هذه الكلمة باليونانية:

الكتابة المغطاة Covered writing.

ويفيد هذا الأسلوب في إخفاء الرسالة وتغطيتها بشيء آخر (ضمن نص آخر أو ضمن صورة). ولا يشكل هذا المجال من التشفير جزءاً من المواضيع التي سنتعرض لها من خلال دراستنا لتقنيات وأساليب التعمية.

الجزء الأول

التشفير بمفتاح متناظر

Symmetric-Key Encipherment

الفصل الأول

تقنيات التشفير متناظرة المفتاح التقليدية

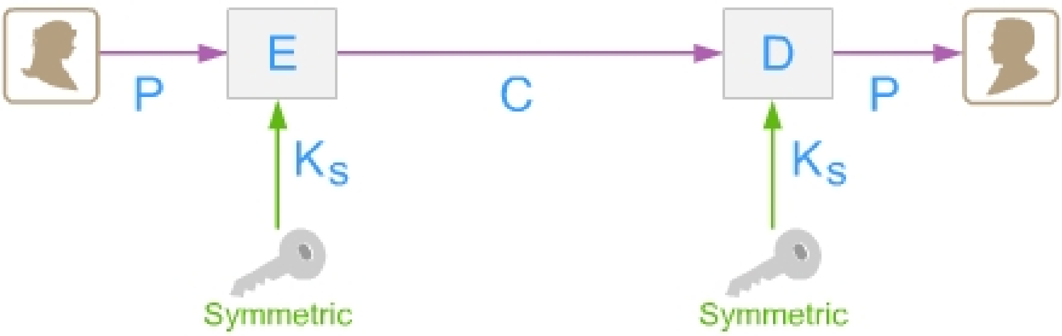
الأهداف Objectives

نستعرض في هذا الفصل أساليب التشفير متناظرة المفتاح التقليدية التي ظهرت واستخدمت في الماضي. سنقوم فيما يلي بشرح أسس هذه الأساليب تمهيداً للحديث فيما بعد عن الأساليب الأخرى الأكثر تطوراً. يهدف هذا الفصل إلى تحقيق ما يلي:

- التعريف بمصطلحات ومفاهيم أساليب التشفير متناظرة المفتاح.
- التركيز على فئتين من أساليب التشفير متناظرة المفتاح التقليدية: الاستبدال **Substitution** والمناقلة **Transposition**.
- التعرض لمفهومي التشفير لدفق المعلومات **Stream cipher** والتشفير الكتلي **Block cipher**.

1. مقدمة Introduction

يبين الشكل التالي المبدأ العام الذي يحكم عمل أساليب التشفير متناظرة المفتاح:



- إذا كان **P** هو النص غير المشفر **Plaintext** و**C** هو النص المشفر **Ciphertext** و**Ks** هو مفتاح التشفير فإن خوارزمية التشفير $E_k(x)$ تعطي النص المشفر من النص غير المشفر وخوارزمية فك التشفير $D_k(x)$ تعطي النص غير المشفر من النص المشفر. نفترض أن $E_k(x)$ و $D_k(x)$ كل واحدة منهما تعمل عكس الأخرى وتقوم كل منهما بإلغاء تأثير الأخرى إذا ما طبقتا على نفس نص الدخل. يكون لدينا بالتالي:

$$Encryption : C = E_k(P)$$

$$Decryption : P = D_k(C)$$

$$D_k(E_k(x)) = E_k(D_k(x)) = x$$

- نستطيع أن نبرهن أن النص غير المشفر الذي أنشأه **Bob** هو نفسه الذي أرسل بالأصل من قبل **Alice**. نفترض أن **Bob** أنشى الملف P_1 ونستطيع أن نبرهن أن $P_1 = P$:

$$Alice : C = E_k(p)$$

$$Bob : P_1 = D_k(C) = D_k(E_k(P)) = P$$

- نحتاج إلى أن نؤكد أن وفقاً لمبدأ كيرشوف (الذي سنتحدث عنه فيما بعد) أنه من الأفضل أن نصرح عن خوارزمتي التشفير وفك التشفير ونجعلهما عامتين وأن نحتفظ بالمفتاح المشترك ونجعله سرياً. يعني ذلك أن **Alice** و**Bob** يحتاجان إلى قناة سرية أخرى من أجل تبادل المفتاح السري. يمكن مثلاً أن:

يلتقيا وجهاً لوجه لمرة واحدة ويتبادلا المفتاح. أو يمكن أن يتم ذلك من خلال طرف ثالث موثوق من قبلهما معاً. كما يمكن أن يتبادلا المفتاح السري باستخدام وسائل تشفير أخرى مثل تلك المعتمدة على المفاتيح غير المتناظرة التي سنتحدث عنها فيما بعد.

- يجب الإشارة هنا إلى أمر هام، وهو عدد المفاتيح المتناظرة التي نحتاج إليها. يجب على **Alice** مثلاً من أجل الاتصال بشخص آخر **David** مثلاً، أن تستخدم مفتاحاً آخراً. بالتالي، إذا كان لدينا m شخص ضمن مجموعة ترغب بالاتصال الآمن بين كل اثنين من أعضائها فإننا نحتاج إلى $(m \times (m-1))/2$ مفتاح لتحقيق ذلك.

مبدأ كيرشوف Kerckhoff's Principle

قد يبدو أن من الآمن أكثر أن نقوم بتخبئة خوارزميتي التشفير وفك التشفير والمفتاح السري كلها دون الإعلان عنها وهو الأمر الذي لا ننصح به.

يجب علينا وفقاً لمبدأ كيرشوف أن نفترض أن الآخرين يعلمون بخوارزمية التشفير أو فك التشفير. وبالتالي تنحصر مهمتنا في مقاومة هجومات فك التشفير في الحفاظ على المفاتيح السرية. سنتبين أهمية هذا المبدأ عندما سنتحدث فيما بعد عن أساليب التشفير الحديثة التي تتمتع بمجال لتوليد المفاتيح واسعاً مما يجعل من الصعب الوصول إليها.

2. التشفير بالاستبدال Substitution Ciphers

يهدف هذا الأسلوب من التشفير إلى استبدال رمز بآخر. إذا كان الرمز هو عبارة عن حرف من الحروف الأبجدية ضمن نص غير مشفر فإننا نقوم باستبداله بحرف آخر. يمكن مثلاً استبدال الحرف **A** بالحرف **D**، والحرف **T** بالحرف **Z** وهكذا. إذا كانت الرموز عبارة عن خانات عددية (من **0** حتى **9**)، فيمكن مثلاً استبدال **3** بـ **7** و **2** بـ **6** وهكذا. يمكن تقسيم طرق الاستبدال ضمن فئتين، هما:

1. المشفرات وحيدة الحرف **Monoalphabetic ciphers**.
2. المشفرات متعددة الحروف **Polyalphabetic ciphers**.

المشفرات وحيدة الحرف Monoalphabetic Ciphers

يجري هنا استبدال حرف (أو رمز) ضمن النص غير المشفر دائماً بحرف (أو رمز) آخر هو نفسه دائماً بغض النظر عن موضعه ضمن النص. أي ان العلاقة بين الحروف ضمن النص غير المشفر والنص المشفر هي علاقة واحد لواحد.

مثال:

نص غير مشفر **Plaintext**: **hello**	نص مشفر **Ciphertext**: **KHOOR**

وتشمل:

1. المشفرات بالجمع.
2. المشفرات بالضرب.
3. المشفرات بالتبديل وحيدة الحرف.

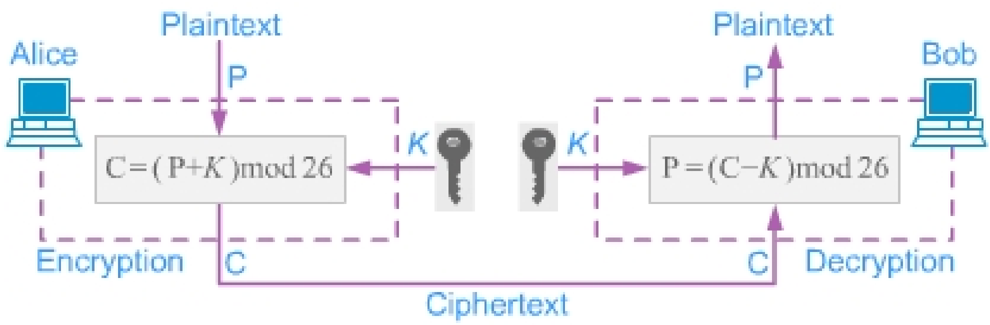
المشفرات بالجمع Additive Ciphers

يعد هذا النوع من المشفرات أبسط مشفرات الاستبدال وحيدة الحرف. وتسمى أحياناً مشفرات الإزاحة **Shift ciphers** أو مشفرات قيصر **Caeser ciphers** ولكن تبقى التسمية مشفرات الجمع أكثر تعبيراً عن الطبيعة الرياضية لهذا النوع من المشفرات. بفرض أن النص غير المشفر يتضمن حرف أبجدية صغيرة (من **a** إلى **z**) وأن النص المشفر يتضمن حروفاً كبيرة (من **A** إلى **Z**)، فبالتالي من أجل تطبيق عمليات حسابية ورياضية على النصين غير المشفر والمشفر يجب أن نخصص قيماً عددية لكل حرف صغير أو كبير كما هو مبين في الشكل التالي:

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

خصص لكل حرف كبير وصغير ضمن الشكل السابق عدداً صحيحاً ضمن المجموعة Z_{26} (وهي مجموعة الأعداد الصحيحة من **0** حتى **25**: $Z_n = \{0,1,2,3,\ldots,n-1\}$). المفتاح الـسري بين **Alice** و**Bob** هو أيضاً عدد صحيح من المجموعة Z_{26}.

تقوم خوارزمية التشفير بإضافة المفتاح إلى حروف النص غير المشفر. أما خوارزمية فك التشفير فتقوم بطرح هذا المفتاح من حروف النص المشفر. تجري كافة العمليات الحسابية ضمن المجموعة Z_{26}. يوضح الشكل التالي كيفية تنفيذ ذلك:



يمكن أن نبرهن بسهولة أن عمليتي التشفير وفك التشفير هما عمليتان متعاكستان وأن النص غير المشفر P_1 والمولد من قبل **Bob** هو نفسه النص P المرسل من قبل **Alice**:

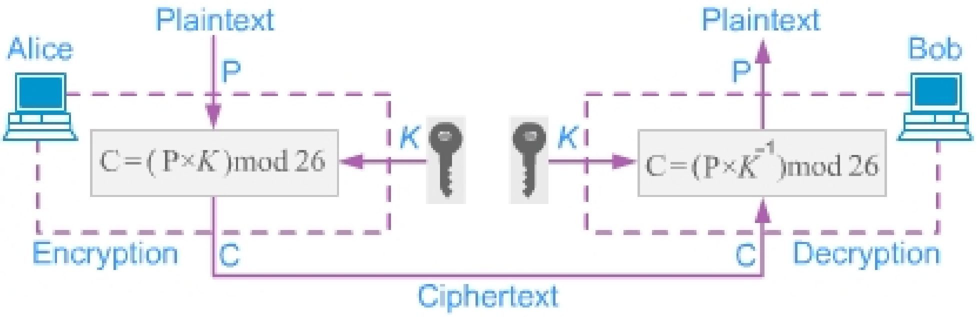
$$P_1 = (C - k)\bmod 26 = (P + k - k)\bmod 26 = P$$

مثال:

K = 3, P	**=**	**RUNAWAY**
E(RUNAWAY)	**→**	**UXQDZDB**
D(UXQDZDB)	**→**	**RUNAWAY**

المشفرات بالضرب Multiplicative Ciphers

تعتمد خوارزميات التشفير هنا على ضرب قيمة حروف النص غير المشفر بقيمة المفتاح للحصول على النص المشفر. أما خوارزميات فك التشفير فتعتمد على قسمة قيم حروف النص المشفر على قيمة المفتاح للحصول على النص غير المشفر، كما هو مبين في الشكل التالي:



يجب أن تجري العمليات الحسابية ضمن المجموعة Z_{26}. كما يجب أن يكون المفتاح عنصراً من عناصر المجموعة Z_{26} التي تضم فقط **12** عنصراً، وهي: **1**، **3**، **5**، **7**، **9**، **11**، **15**، **17**، **19**، **21**، **23**، **25**.

ملاحظة:

المجموعة $Z_{26}*$ هي مجموعة جزئية من Z_{26} وتضم فقط الأعداد التي لها أزواج جداء عكسية، أي أزواج التي يكون نتيجة جدائها مساوي للقيمة **1**، كما هو مبين في الجدول التالي:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	0	2	4	6	8	10	12	14	16	18	20	22	24	0	2	4	6	8	10	12	14	16	18	20	22	24
3	0	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2	5	8	11	14	17	20	23
4	0	4	8	12	16	20	24	2	6	10	14	18	22	0	4	8	12	16	20	24	2	6	10	14	18	22
5	0	5	10	15	20	25	4	9	14	19	24	3	8	13	18	23	2	7	12	17	22	1	6	11	16	21
6	0	6	12	18	24	4	10	16	22	2	8	14	20	0	6	12	18	24	4	10	16	22	2	8	14	20
7	0	7	14	21	2	9	16	23	4	11	18	25	6	13	20	1	8	15	22	3	10	17	24	5	12	19
8	0	8	16	24	6	14	22	4	12	20	2	10	18	0	8	16	24	6	14	22	4	12	20	2	10	18
9	0	9	18	1	10	19	2	11	20	3	12	21	4	13	22	5	14	23	6	15	24	7	16	25	8	17
10	0	10	20	4	14	24	8	18	2	12	22	6	16	0	10	20	4	14	24	8	18	2	12	22	6	16
11	0	11	22	7	18	3	14	25	10	21	6	17	2	13	24	9	20	5	16	1	12	23	8	19	4	15
12	0	12	24	10	22	8	20	6	18	4	16	2	14	0	12	24	10	22	8	20	6	18	4	16	2	14
13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13	0	13
14	0	14	2	16	4	18	6	20	8	22	10	24	12	0	14	2	16	4	18	6	20	8	22	10	24	12
15	0	15	4	19	8	23	12	1	16	5	20	9	24	13	2	17	6	21	10	25	14	3	18	7	22	11
16	0	16	6	22	12	2	18	8	24	14	4	20	10	0	16	6	22	12	2	18	8	24	14	4	20	10
17	0	17	8	25	16	7	24	15	6	23	14	5	22	13	4	21	12	3	20	11	2	19	10	1	18	9
18	0	18	10	2	20	12	4	22	14	6	24	16	8	0	18	10	2	20	12	4	22	14	6	24	16	8
19	0	19	12	5	24	17	10	3	22	15	8	1	20	13	6	25	18	11	4	23	16	9	2	21	14	7
20	0	20	14	8	2	22	16	10	4	24	18	12	6	0	20	14	8	2	22	16	10	4	24	18	12	6
21	0	21	16	11	6	1	22	17	12	7	2	23	18	13	8	3	24	19	14	9	4	25	20	15	10	5
22	0	22	18	14	10	6	2	24	20	16	12	8	4	0	22	18	14	10	6	2	24	20	16	12	8	4
23	0	23	20	17	14	11	8	5	2	25	22	19	16	13	10	7	4	1	24	21	18	15	12	9	6	3
24	0	24	22	20	18	16	14	12	10	8	6	4	2	0	24	22	20	18	16	14	12	10	8	6	4	2
25	0	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

مثال:

K = 7, P	**=**	**hello**
E(hello)	**→**	**XCZZU**
D(XCZZU)	**→**	**hello**

المشفرات بالتبديل وحيدة الحرف Monoalphabetic Substitution Ciphers

على اعتبار أن المشفرات بالجمع وبالضرب لديها مجال صغير لاختيار المفتاح المرتبط بها، فإنه من الممكن أن يكون من السهل كسرها. يمكن أن نوفر حلاً أفضل بأن يتفق كلا الطرفين على جدول للمطابقة بين الحروف المتبادلة. يعرض الشكل التالي مثالاً عن هذا الجدول:

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

مثال:

"runaway" → "HJGNPNS"

المشفرات متعددة الحروف Polyalphabetic Ciphers

- مع هذا النوع من المشفرات، يمكن أن يتوفر عدة خيارات لاستبدال كل حرف من الحروف. فالعلاقة بين أي حرف من حروف النص غير المشفر مع أي حرف ضمن النص المشفر هي علاقة حرف واحد مع أكثر **one-to-many**. يمكن مثلاً استبدال الحرف **"a"** بالحرف **D** إذا ورد في بداية النص أو بالحرف **N** إذا ورد في منتصفه.
- يضمن هذا النوع من المشفرات إخفاء تواتر ورود الحروف ضمن النص. وبالتالي تصبح محاولات كسر التشفير المعتمدة على تواتر ورود الحروف ضمن النص غير فعالة.
- لبناء مشفر متعدد الحروف، يجب الربط بين الحرف ضمن النص المشفر والحرف المقابل له ضمن النص غير المشفر ومكان تواجده ضمن هذا النص. هذا يعني أن المفتاح في هذه الحالة، هو عبارة عن مجموعة من المفاتيح الجزئية. يرتبط كل واحد منها بموقع الحرف ضمن النص غير المشفر. بمعنى آخر:

فإن المفتاح $k = (k_1, k_2, k_3, \ldots)$ يفيد أنه يمكن استخدام k_i لتشفير الحرف رقم **i** ضمن النص غير المشفر لبناء الحرف رقم **i** ضمن النص المشفر.

أمثلة:

- **التشفير بمفتاح ذاتي التوليد Autokey Cipher**: يتكون المفتاح مع هذا النوع من المشفرات من مجموعة من المفاتيح الجزئية متتابعة حيث يجري استخدام كل مفتاح جزئي لتشفير الحرف الموافق لترتيب المفتاح ضمن النص غير المشفر. يجري الاتفاق حول قيمة المفتاح الجزئي الأول. ويكون المفتاح الجزئي الثاني هو قيمة الحرف الأول ضمن النص غير المشفر. أما المفتاح الجزئي الثالث، فهو قيمة الحرف الثاني ضمن النص غير المشفر وهكذا دواليك.

$$P = P_1 P_2 P_3 \cdots$$
$$C = C_1 C_2 C_3 \cdots$$
$$k = (k_1, P_1, P_2, \cdots)$$
$$Encryption : C_i = (P_i + k_i) \bmod 26$$
$$Decryption : P_i = (C_i - k_i) \bmod 26$$

أمثلة:

- **المشفر فيجينر Vigenere Cipher**: جرى تصميم هذا المشفر من قبل الرياضي الفرنسي فيجينر في القرن السادس عشر. يعتمد هذا المشفر على تكرار استخدام مفتاح مع دفق حروف النص غير المشفر بشكل متتالي. يجب أن يكون طول المفتاح **m** محقق للشرط $1 \le m \le 26$.

$$P = P_1 P_2 P_3 \cdots$$
$$C = C_1 C_2 C_3 \cdots$$
$$K = [(k_1, k_2, \ldots, k_m), (k_1, k_2, \ldots, k_m), \cdots]$$
$$Encryption : C_i = (P_i + k_i) \bmod 26$$
$$Decryption : P_i = (C_i - k_i) \bmod 26$$

مثال:

Key: “python”

Plaintext: “rabbitwithbigpointyteeth”

Ciphertext:

R	a	b	b	i	t	w	i	t	h	b	i	g	p	o	i	n	t	y	t	e	e	t	H
p	y	t	h	o	n	p	y	t	h	o	n	p	y	t	h	o	n	p	y	t	h	o	N
G	Y	U	I	V	G	L	G	M	Y	M	V	V	N	H	P	B	G	N	R	P	L	H	U

أمثلة:

- **المشفر لمرة واحدة One-time Pad**: يكون في هذا المشفر طول المفتاح السري العشوائي مساوياً لطول النص غير المشفر. مما يجعل من المستحيل عملياً كسر هذا المشفر ومعرفة النص المرسل في الأصل وفقاً لما جاء في دراسة قام بها العالم شانون بهذا الخصوص. تزداد سرية هذا المشفر عند تغيير المفتاح عند تغيير النص المرسل حيث لا يجري استخدامه إلا مرة واحدة فقط. يؤدي ذلك إلى ضمان السرية بشكل كبير ولكن يستحيل عملياً تحقيق ذلك بشكل سهل وتبادل المفتاح من وقت إلى آخر بين الطرفين المتصلين.

مثال:

Ciphertext: NZAKBMK

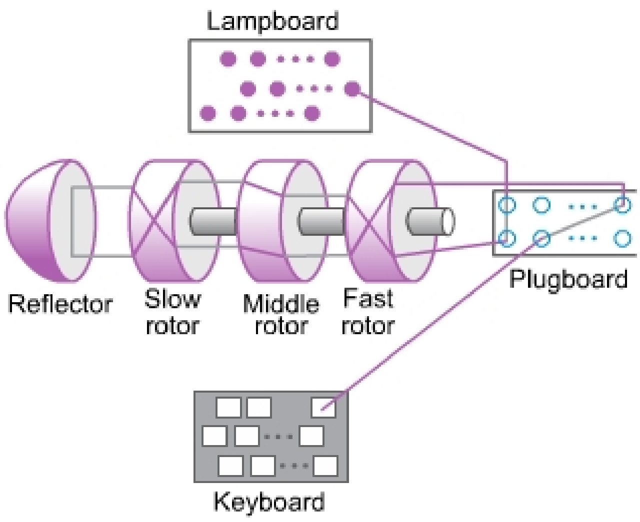
Possible Vigenére keys: wtnkxmm and nlvwker

Ciphertext:	**NZAKBMK**	**NZAKBMK**
Possible keys:	**nlvwker**	**wtnkxmm**
Plaintext:	**goforit**	**runaway**

ما هو النص الصحيح الذي تم ارساله؟

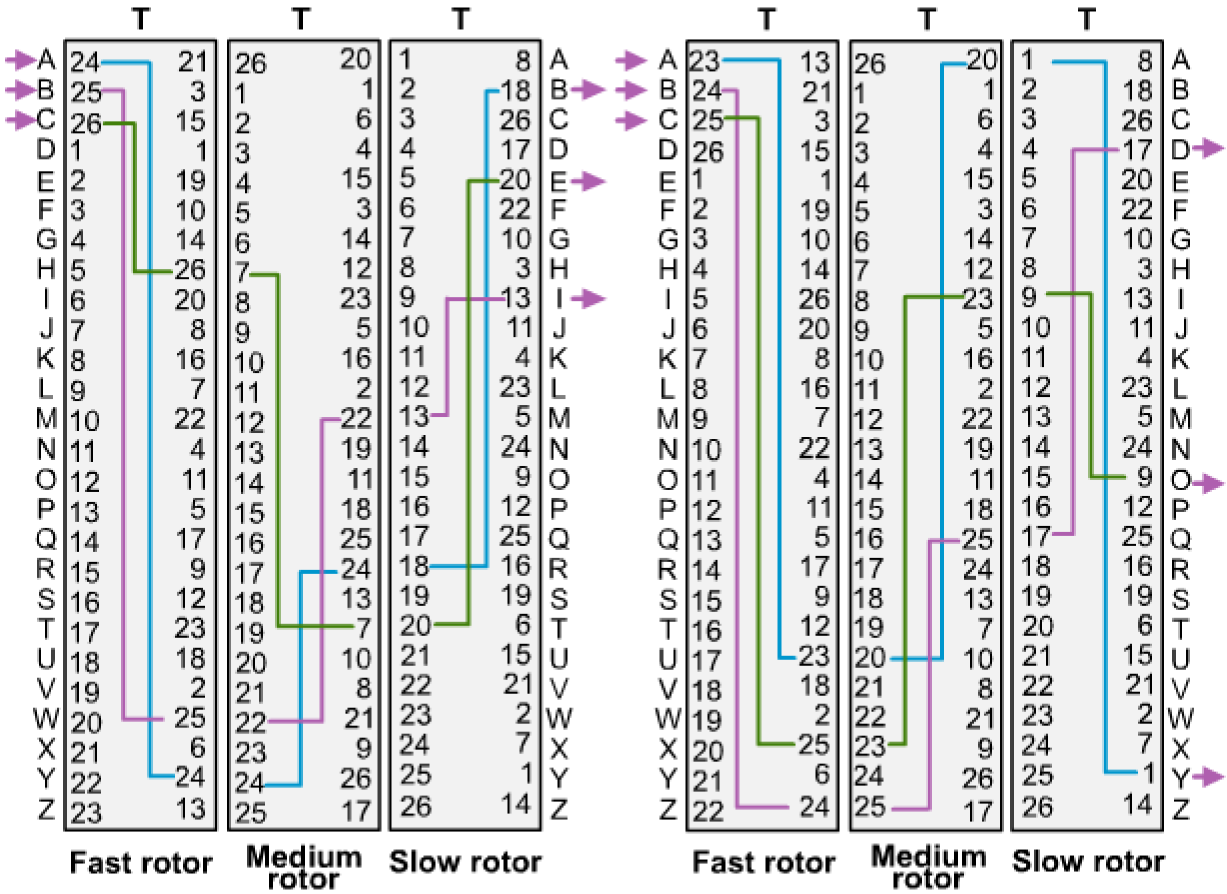
أمثلة:

- **آلة انيغما Enigma Machine**: جرى تطوير هذه الآلة من قبل الألمان أثناء الحرب العالمية الثانية. وهي آلة معقدة تعمل وفقاً لمبدأ التشفير بالتبديل. يبين الشكل التالي منظر الآلة التي سنشرح فيما يلي طريقة عملها:



- تتضمن الآلة ثلاثة دواليب من بين خمسة يمكن تركيبها. تعمل هذه الدواليب على الشكل التالي:
  1. يقوم كل دولاب بعملية تشفير بالتبديل وحيدة الحرف.
  2. يجري تمرير نتيجة تشفير كل دولاب إلى الدولاب التالي ليجري تشفيرها مرة أخرى وهكذا وصولاً إلى الدولاب الثالث.
  3. تدور الدواليب من أجل كل حرف من حروف النص غير المشفر وفقاً لما يلي:
     - الدولاب السريع **Fast rotor**: من أجل كل حرف.
     - الدولاب المتوسط **Middle rotor**: من أجل كل **26** حرف.
     - الدولاب البطيء **Slow rotor**: من أجل كل **676 = 26 x 26** حرف.
- إذا جرى إدخال الحرف **C** على لوحة المفاتيح كحرف أول فإنه سيجري ما يلي:
  1. سيطابق هذا الحرف الرقم **26** على الدولاب السريع.
  2. الرقم **26** مرتبط بالرقم **7** على الدولاب الوسيط.
  3. الرقم **7** مرتبط بالرقم **20** على الدولاب البطيء.
  4. يوافق الرقم **20** الحرف **E** كنتيجة لعملية التشفير.
- إذا جرى إدخال الحرف **C** على لوحة المفاتيح كحرف ثاني فإنه سيجري ما يلي بعد دوران الدولاب السريع لحرف واحد:
  1. سيطابق هذا الحرف الرقم **25** على الدولاب السريع.
  2. الرقم **25** مرتبط بالرقم **23** على الدولاب الوسيط.
  3. الرقم **23** مرتبط بالرقم **9** على الدولاب البطيء.
  4. يوافق الرقم **9** الحرف **O** كنتيجة لعملية التشفير.

• يجري تكرار التشفير للحروف بعد **<u>17,576</u> = 26 x 26 x 26** حرف.



3. التشفير بالمناقلة Transposition Ciphers

تعتمد هذه المشفرات على فكرة مكان تواجد الحروف والرموز ضمن النص غير المشفر بدلاً من استبدالها. فيمكن مثلاً لحرف متواجد ضمن الموضع الأول ضمن النص غير المشفر أن يظهر في الموضع العاشر في النص المشفر وهكذا بالنسبة لبقية الحروف.

وتشمل:

1. مشفرات المناقلة عديمة المفتاح.
2. مشفرات المناقلة المزودة بمفتاح.
3. الدمج بين الأسلوبين السابقين.

مشفرات المناقلة عديمة المفتاح Keyless Transposition Ciphers

وهي من أبسط مشفرات المناقلة حيث يجري معها تدوير الحروف وفقاً لطريقتين، هما:

في الطريقة الأولى: يكتب النص غير المشفر ضمن جدول بحيث تملئ خانات الجدول عموداً فعمود ويجري ارساله سطراً فسطر.

في الطريقة الثانية: فيكتب النص غير المشفر ضمن جدول بحيث تملئ خانات الجدول سطراً فسطر ويجري ارساله عموداً فعمود.

مثال:

يمكن للطرفين أن يتفقا حول عدد الأعمدة ضمن الجدول الذي يجري ملئه قبل ارسال الرسالة.

ثم يجري نقل الرسالة عمود فعمود. أما الطرف المستقبل للرسالة فإنه يتبع الخطوات المعاكسة لعملية الإرسال لفك تشفير الرسالة، حيث يقوم بكتابتها ضمن الجدول عمود فعمود، ثم يقرأها سطراً فسطر.

لنحصل على الرسالة المشفرة **"MMTAEEHREAEKTTP"**

m	E	e	T
m	E	a	T
t	H	e	P
a	R	k	

مشفرات المناقلة المزودة بمفتاح Keyed Transposition Ciphers

تعتمد مشفرات المناقلة بدون مفتاح على فكرة كتابة النص غير المشفر بطريقة وقراءته بطريقة أخرى. تجري عملية المناقلة على كامل النص غير المشفر لخلق النص المشفر. يمكن أيضاً تقسيم النص غير المشفر إلى أجزاء من حجم محدد (تسمى بالكتل **Blocks**) واستخدام مفتاح محدد لمناقلة الحروف ضمن كل كتلة بشكل منفصل عن الكتل الأخرى.

مثال:

يمكن لطرفي الاتصال أن يتفقا على تقسيم النص إلى أجزاء وكتل من خمس حروف والمناقلة فيما بينها وفقاً لمفتاح من الممكن استخدامه في التشفير وفي فك التشفير أيضاً، كما هو مبين في المفتاح التالي:

Encryption التشفير	↓	3	1	4	5	2	↑	**Decryption** فك التشفير
		1	2	3	4	5		

يجري مناقلة الحرف الثالث ضمن النص غير المشفر إلى موضع الحرف الأول ضمن الكتلة المولدة من النص المشفر. وبالعكس الحرف الأول من كتلة النص المشفر سيصبح موضع الحرف الثالث من النص غير المشفر وهكذا دواليك.

بموجب الفرضيات السابقة، إذا كان لدينا النص التالي **"Enemy attacks tonight"** فإننا نتعامل معه على الشكل التالي:

- كتل النص غير المشفر:

ightz	kston	attac	enemy

- كتل النص المشفر:

HITZG	TKONS	TAACT	EEMYN

لنحصل على النص المشفر التالي **"EEMYNTAACTTKONSHITZG"**.

الدمج بين الأسلوبين السابقين Combining Two Approaches

تعتمد طرق التشفير بالمناقلة الحديثة على الدمج بين الأسلوبين السابقيين، حيث يتم التشفير وفك التشفير على ثلاث خطوات:

- يجري في الخطوة الأولى كتابة النص ضمن جدول محدد سطراً فسطر.
- نقوم في الخطوة الثانية بمناقلة الأعمدة فيما بينها وفقاً لمفتاح متفق عليه.

- ثم نقوم في الخطوة الأخيرة بقراءة النص عموداً فعموداً.

4. التشفير لدفق المعطيات والتشفير الكتلي Stream and Block Ciphers:

جرى ضمن المراجع العلمية للمشفرات ذات المفاتيح المتناظرة تقسيم هذه المشفرات ضمن فئتين رئيسيتين، هما:

مشفرات دفق المعطيات Stream Ciphers:

يجري مع هذه المشفرات التعامل مع المعطيات على الرمز الواحد (أو الحرف الواحد) في نفس الوقت. ويكون لدينا بالتالي: دفق لمعطيات النص غير المشفر، دفق لمعطيات النص المشفر، دفق لمعطيات مفتاح التشفير:

$$
\begin{aligned}
P &= P_1P_2P_3\cdots \\
C &= C_1C_2C_3\cdots \\
K &= (k_1,k_2,k_3,\cdots) \\
C_1 &= E_{k1}(P_1) \\
C_2 &= E_{k2}(P_2) \\
C_3 &= E_{k3}(P_3)\cdots
\end{aligned}
$$

المشفرات الكتلية Block Ciphers:

يجري مع هذه المشفرات تقسيم النص غير المشفر إلى كتل من الحجم $m \succ 1$ ليتم بعد ذلك تشفيرها معاً وإنشاء كتلة من نفس الحجم ضمن النص المشفر. يجري مع هذه المشفرات استخدام مفتاح واحد من كامل الكتلة حتى ولو كان هذا المفتاح مؤلفاً من عدة قيم جزئية.

الفصل الثاني

تقنيات التشفير متناظرة المفتاح الحديثة

Modern Symmetric-Key

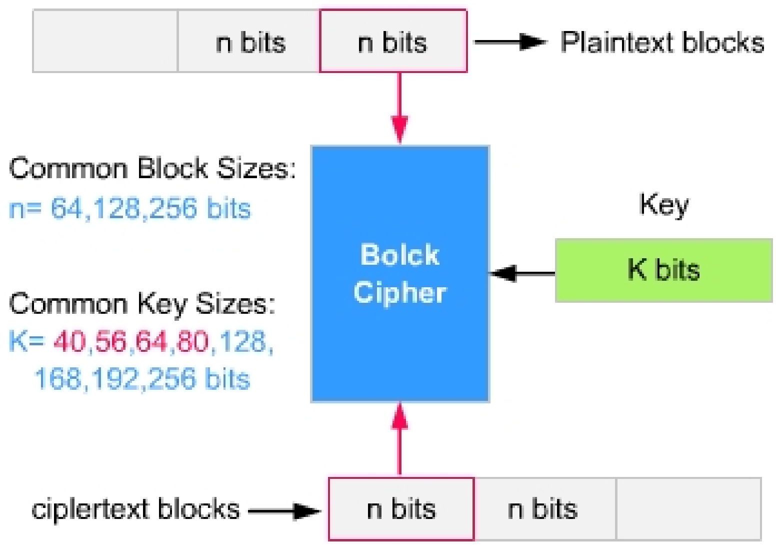
الأهداف Objectives

يهدف هذا الفصل إلى التعريف بما يلي:

- كيف يمكن التمييز بين أساليب التشفير متناظرة المفتاح التقليدية والحديثة.
- أساليب التشفير الكتلية الحديثة وميزاتها ومكوناتها.
- مشفرات الجداء **Product ciphers**.
- مشفرات دفق المعلومات المتزامنة وغير المتزامنة وكيفية بناءها وتحقيقها.

1. المشفرات الكتلية الحديثة Modern Block Ciphers

تعتمد هذه المشفرات على التعامل مع كتل بحجم **n** بت. وتستخدم خوارزميات التشفير وفك التشفير الخاصة بها مفتاح سري مشترك بطول **k** بت، كما هو مبين في الشكل التالي حيث يجري التعامل مع ملفات ثنائية **Binary files**:



إذا تضمن الرسالة نصاً ذو حجم أقل من **n** بت، يجري عندها إضافة بتات حشو إلى النص ليصل إلى الحجم المطلوب.

وتشمل:

1. الاستبدال أو المناقلة **Substitution or Transposition**.
2. المشفرات الكتلية وزمر التدوير **Block Ciphers as Permutation Groups**.
3. مكونات المشفر الكتلي الحديث **Components of a Modern Block Cipher**.

الاستبدال أو المناقلة Substitution or Transposition

يمكن استخدام أحد أسلوبي الاستبدال أو المناقلة مع المشفرات الكتلية الحديثة بنفس طريقة عمل المشفرات التقليدية، لكن نقوم مع المشفرات الحديثة بالتعامل مع بتات وليس مع رموز أو حروف.

إذا كان المشفر معتمد على أسلوب الاستبدال فإنه بتبديل بت قيمتها **1** أو **0** ضمن النص غير المشفر ببت قيمتها **0** أو **1** يعني ذلك أن النصين غير المشفر والمشفر سيكون لديهما عدد مختلف من بتات **1** و**0**. أما إذا كان المشفر معتمد على أسلوب المناقلة، فإننا سنقوم بتغيير مواضع البتات فقط وسنحصل على نفس عدد بتات **1** و**0** ضمن النصين غير المشفر والمشفر.

في كلا الحالتين السابقتين، فإن القيم المحتملة لكتلة من **n** بت هي 2^n قيمة، لأن كل بت يمكن أن تأخذ أحد قيمتين: إما **0** أو **1**.

جرى تصميم المشفرات الكتلية الحديثة على أنها مشفرات استبدال لأن خواص المناقلة التي تؤدي إلى المحافظة على عدد خانات **0** والـ **1**، تجعل المشفر قابل للكسر بسهولة.

المشفرات الكتلية وزمر التدوير Block Ciphers as Permutation Groups

يمكن أثناء دراستنا للمشفرات الكتلية الحديثة أن نسأل فيما إذا كان يمكن النظر إلى المشفر الكتلي الحديث على أنه زمرة رياضية **Group**؟

للإجابة على هذا السؤال، يجب أن نفترض:

أولاً أن مفتاح التشفير طويل بشكل كافٍ لتحديد الخيار المناسب من بين الخيارات الممكنة التي تربط بين الدخل والخرج.

نسمي المشفر الذي يتمتع بهذه الصفة بالمشفر ذو المفتاح الكامل **Full-size key cipher**.

غالباً ما يكون المفتاح عملياً أصغر وبالتالي تصبح بعض الخيارات للمطابقة بين الدخل والخرج ممكنة فقط وليس كلها.

كما يجب ملاحظة أيضاً أنه على الرغم من أن المشفرات الكتلية تعتمد على وجود مفتاح سري مشترك بين طرفي تبادل المعلومات، إلا أنه يوجد داخل المشفر مكونات لا تعتمد على وجود مفتاح.

وتشمل:

1. المشفرات الكتلية بالمناقلة ذات المفتاح الكامل.

2. المشفرات الكتلية بالاستبدال ذات المفتاح الكامل.
3. زمرة التدوير المرتبطة بمشفرات المفتاح الكامل.
4. المشفرات ذات المفتاح الجزئي.
5. زمرة التدوير المرتبطة بمشفرات المفتاح الجزئي.
6. المشفرات عديمة المفتاح.

المشفرات الكتلية بالمناقلة ذات المفتاح الكامل Full-Size Key Transposition Block Ciphers:

تقوم هذه المشفرات بتغيير موضع الخانات دون تغيير قيمتها. أي محاكاة عملها كما لو أنه اختيار لعنصر من **n** خانة من أحد جداول التدوير **Permutation Tables** التي يبلغ عددها **n!** جدول وذلك وفقاً للمفتاح الذي يحدد الجدول المناسب. نحتاج إذاً أن يتوفر لدينا **n!** مفتاح ممكن ويجب أن يضم كل مفتاح $\lceil \log_2 n! \rceil$ خانة.

المشفرات الكتلية بالاستبدال ذات المفتاح الكامل Full-Size Key Substitution Block Ciphers:

لا تقوم هذه المشفرات بتغيير موضع الخانات وإنما باستبدال قيمتها. قد يبدو للوهلة الأولى أنه لا يمكن محاكاة عملها كما لو أنه اختيار لعملية تدوير **Permutation** ممكنة. لكن ذلك ممكن إذا كان بالإمكان فك كود **Decoding** الدخل وتكوين الخرج **Encoding**.

نعني بعملية فك الكود **Decoding**:

تحويل الأعداد الصحيحة المؤلفة من **n** خانة إلى سلسلة طولها 2^n خانة مؤلفة من خانة واحدة ذات قيمة مساوية لـ **1** و 2^n-1 خانة مساوية لـ **0**.

يعبر موضع الخانة **1** عن قيمة العدد الصحيح الذي يمكن أن يأخذ أحد القيم من **0** إلى 2^n-1. أما عملية التكويد **Encoding**، فتعني العملية العكسية لما سبق. وعلى اعتبار أن الدخل والخرج الجديدين يتضمنان دائماً قيمة خانة **1** وحيدة، فإنه يمكن النظر إلى هذا النوع من المشفرات على أنها تدوير ضمن $2^n!$ عنصر ممكن.

زمرة التدوير المرتبطة بمشفرات المفتاح الكامل Permutation Group for full-size key siphers:

على اعتبار أن المشفرات الكتلية الحديثة بالاستبدال أو المناقلة ذات المفتاح الكامل هي عمليات تدوير، فإنه إذا جرى التشفير (أو فك التشفير) أكثر من مرحلة فإن النتيجة تكون مكافئة إلى زمرة تدوير وفقاً لعملية التركيب **Composition Operation**.

أي يمكن الاستعاضة عن عمليتي تدوير متتاليتين أو أكثر بعملية تدوير واحدة. هذا يؤكد أنه من غير الضروري وجود أكثر من مرحلة تشفير بمفتاح كامل لأنها ستكون مكافئة لمرحلة واحدة بالنتيجة.

المشفرات ذات المفتاح الجزئي Partial-Size Key Ciphers:

لا يمكن للمشفرات الفعلية أن تستخدم مفاتيح كاملة لأنها ستكون طويلة جداً، خاصةً إذا كانت هذه المشفرات هي مشفرات كتلية بالاستبدال. من بين المشفرات الكتلية بالاستبدال لدينا مثلاً المشفر **DES** (الذي سنتحدث عنه لاحقاً) الذي يعتمد على تشفير كتل من حجم **64** بت فقط.

أي أننا نحتاج من أجل هذه الكتل إلى مفتاح كامل بطول $\log_2(2^{64}!) \approx 2^{70}$ خانة. لكن عملياً، يبلغ طول مفتاح هذا المشفر **56** بت فقط. يعني ذلك أنه لدينا فقط 2^{56} خيار من بين $2^{2^{70}}$ خيار ممكن المطابقة معها.

زمرة التدوير المرتبطة بمشفرات المفتاح الجزئي Permutation Group for partial-size key siphers:

السؤال الذي يطرح نفسه الآن، هل تشكل مراحل المناقلة أو الاستبدال المتعددة باستخدام مفتاح جزئي زمرة تدوير وفقاً لعملية التركيب **Composition Operation**؟

الإجابة على هذا السؤال مهمة لأنها تعلمنا فيما إذا كانت المراحل المتعددة والمتتالية من المشفر يمكن أن تؤدي إلى تحقيق مستوى تشفير أعلى (كما سنلاحظ أثناء الحديث عن المشفر **DES** لاحقاً).

يمكن أن يشكل المشفر المزود بمفتاح جزئي زمرة **Group** إذا كان يشكل زمرة جزئية **Subgroup** من المشفر المزود بمفتاح كامل موافق له. على سبيل المثال:

جرى البرهان على أن عدة مراحل من المشفر **DES** المزود بمفتاح طوله **56** بت لا يشكل زمرة لأنه لا تشكل الخيارات الممكنة له (وهي 2^{56} خيار للمطابقة معها) زمرة جزئية من زمرة الخيارات الكلية (والتي يبلغ عدد عناصرها $2^{64}!$ خيار).

المشفرات عديمة المفتاح Keyless Ciphers:

على الرغم من أن المشفرات عديمة المفتاح ليست مفيدة بحد ذاتها إلا أنها مستخدمة ضمن مكونات المشفرات ذات المفاتيح:

- نسمي المشفرات عديمة المفتاح التي تعتمد على أسلوب المناقلة بـ "صناديق التدوير **P-boxes**" والتي سنتحدث عنها فيما بعد. يمكن تخيلها كما لو أنها مكونات الصلبة للمناقلة المسبق تعريفها أو جدول التدوير للإمكانيات الممكن الإختيار من بينها لتحقيق المناقلة إذا جرى تحقيقها برمجياً.
- نسمي المشفرات عديمة المفتاح التي تعتمد على أسلوب الاستبدال بـ "صناديق الاستبدال **S-boxes**" والتي سنتحدث عنها فيما بعد. يمكن تخيلها كما لو أنها مكونات الصلبة للمطابقة المسبق تعريفها أو جدول أو تابع رياضي لحساب الإختيار المطلوب لتحقيق الاستبدال إذا جرى تحقيقها برمجياً.

1. المشفرات الكتلية الحديثة Modern Block Ciphers

مكونات المشفر الكتلي الحديث Components of a Modern Block Cipher

المشفر الكتلي الحديث هو عادةً مشفر كتلي بالاستبدال مزود بمفتاح.

يسمح هذا المفتاح بمطابقة جزئية بين الإمكانات المتاحة للدخل مع الإمكانات المتاحة للخرج. يضاف إلى ما سبق، أن هذا النوع من المشفرات ليس مصمماً على شكل وحدة واحدة، وإنما جرى بناؤه على شكل تجميع لوحدات للمناقلة (والتي نسميها صناديق التدوير **P-boxes**) ووحدات الاستبدال (والتي نسميها صناديق الاستبدال **S-boxes**)، إضافة إلى وحدات أخرى.

وتشمل:

1. صناديق التدوير.
2. صناديق الاستبدال.
3. مشفرات الجداء.
4. صفوف مشفرات الجداء.

صناديق التدوير P-Boxes

يهدف صندوق التدوير إلى تبديل موضع الخانات. يوجد ثلاثة أنواع منها: المباشرة **Straight**، والموسعة **Expansion**، والمضغوطة **Compression**:

- **صناديق التدوير المباشرة**: وهي صناديق لها n مدخل وn مخرج لإمكانية تدوير ممكنة. يوجد مع هذه الصناديق $n!$ إمكانية تدوير ممكنة.
- **صناديق التدوير المضغوطة**: وهي صناديق لها n مدخل وm مخرج لإمكانية تدوير ممكنة ($m \prec n$). يجري هنا إيقاف بعض المداخل بحيث لا تصل إلى مخارج الصندوق. تستخدم هذه الصناديق لتنفيذ عملية تدوير واستبعاد عدد من الخانات من المرحلة التالية من التشفير.
- **صناديق التدوير الموسعة**: وهي صناديق لها n مدخل وm مخرج لإمكانية تدوير ممكنة ($m \succ n$). يجري هنا تكرار وصل بعض المداخل مع أكثر من مخرج.. تستخدم هذه الصناديق لتنفيذ عملية تدوير وزيادة عدد الخانات كدخل للمرحلة التالية من التشفير.
- **قابلية العكس Invertibility**: تعتبر صناديق التدوير المباشرة صناديق قابلة للعكس، أي تمكن استخدامها لتنفيذ أعمال التشفير وفك التشفير بعد عكس محتوى جداول التدوير. أما بالنسبة لصناديق التدوير المضغوطة والموسعة فلا يمكن عكسها. فيمكن مثلاً في صندوق التدوير المضغوط تجاهل أحد المداخل أثناء عملية التشفير ولكن لا يوجد أية طريقة للتعويض عنه أثناء عملية فك التشفير. كذلك الأمر مع صناديق التدوير الموسعة.

صناديق الاستبدال S-Boxes

يمكن التفكير بها كما لو أنها مشفرات تبديل مصغرة. يمكن لهذه الصناديق أن يكون لها عدد مختلف من المداخل والمخارج. يمكن أيضاً أن تكون مزودة بمفتاح أو عديمة المفتاح، لكن تستخدم المشفرات الكتلية الحديثة صناديق الاستبدال عديمة المفتاح ذات المطابقة بين المداخل والمخارج المعرفة مسبقاً:

1. صناديق الاستبدال الخطية.
2. قابلية العكس.
3. التوابع الثنائية.

صناديق الاستبدال S-Boxes

صناديق الاستبدال الخطية Linear S-boxes

بفرض أنه كان لدينا ضمن صندوق استبدال المداخل التالية: $x_o, x_1, \ldots, x_n$ والمخارج التالية: $y_o, y_1, \ldots, y_m$ وكان يمكن تمثيل العلاقة بين المداخل والمخارج على شكل مجموعة من المعادلات كما يلي:

$$y_1 = f_1(x_1, x_2, \ldots, x_n)$$
$$y_2 = f_2(x_1, x_2, \ldots, x_n)$$
$$\ldots$$
$$y_m = f_m(x_1, x_2, \ldots, x_n)$$

فيمكن عندها التعبير عن صندوق الاستبدال الخطي على الشكل التالي:

$$y_1 = a_{1,1}x_1 \oplus a_{1,2}x_2 \oplus \cdots \oplus a_{1n}x_n$$
$$y_2 = a_{2,1}x_1 \oplus a_{2,2}x_2 \oplus \cdots \oplus a_{2,n}x_n$$
$$\ldots$$
$$y_m = a_{m,1}x_1 \oplus a_{m,2}x_2 \oplus \cdots \oplus a_{m,n}x_n$$

أما بالنسبة لصناديق الاستبدال غير الخطية فلا يمكن ايجاد مثل هذه العلاقة.

قابلية العكس Invertibility

يمكن لصناديق الاستبدال قابلة للعكس إذا كان عدد المداخل مساوياً لعدد المخارج.

التوابع الثنائية Binary Functions

على اعتبار النص غير المشفر والمفتاح مؤلفين من قيم ثنائية فيمكن استخدام توابع ثنائية للتشفير وفك التشفير.

مثال1:

استخدام التابع AND

- **Plaintext:** 1001101110101100
- **Key:** <u>1101100011001010</u>
- **Ciphertext:** 1001100010001000

تكمن المشكلة في التعامل مع بعض التوابع الثنائية في كونها غير قابلة للعكس. التابع AND غير قابل للعكس فلا يمكن باستخدامه استنتاج النص غير المشفر في حال معرفة المفتاح والنص المشفر، كما هو مبين في المثال التالي:

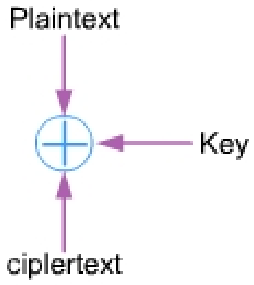
- **Plaintext:** يمكن أن تكون 0 أو 1 ← ?
- **Key:** <u>0</u>
- **Ciphertext:** 0

التوابع الثنائية Binary Functions

مثال2:

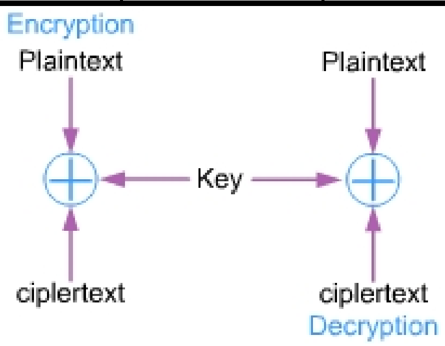
استخدام التابع XOR (Exclusive OR) الذي يمكن تعريفه واستخدامه للتشفير كما هو مبين في الشكل التالي:

P	K	C = P ⊕ K
1	1	0
1	0	1
0	1	1
0	0	0



يتمتع هذا التابع بإمكانية عكس ذاته، أي أن: $C = P \oplus K \rightarrow P = C \oplus K$.

C = P ⊕ K	K	P must be:
1	1	0
1	0	1
0	1	1
0	0	0



تستخدم المشفرات الكتلية الحديثة التابع **XOR** للتشفير كما هو مبين في المثال التالي الذي يتعامل مع كتل مكونة من **8** خانات:

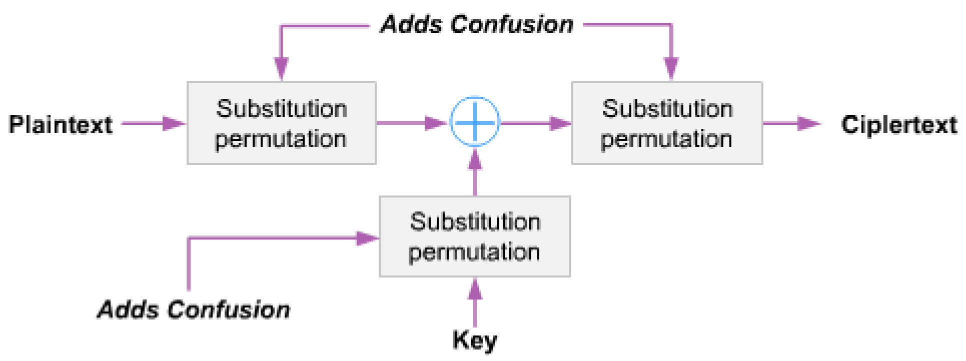
Encryption:			
Plaintext:	**10010101**	**00100110**	**01110101**
Key:	**10100110**	**10100110**	**10100110**
Ciphertext:	**00110011**	**10000000**	**01010011**
Decryption:			
Ciphertext:	**00110011**	**10000000**	**01010011**
Key:	**10100110**	**10100110**	**10100110**
Plaintext:	**10010101**	**00100110**	**01110101**

ملاحظة هامة: إذا جرى استخدام التابع **XOR** لوحده فإن عملية كسر التشفير تكون سهلة جداً ويمكن إيجاد المفتاح بواسطة أول نص غير مشفر معلوم وفقاً للمعادلة التالية: $K = P \oplus C$. لذلك يجب اللجوء إلى إضافة أساليب تمويه أخرى والتي نذكر منها الأسلوبين: البث **Diffusion** والخلط **Confusion** والتي سنتحدث عنهما فيما يلي.

مشفرات الجداء Product Ciphers

وهي مشفرات معقدة مكونة من عدة مكونات للاستبدال والتدوير وغير ذلك من المكونات الأخرى. تعتمد هذه المكونات على أسلوبين للتمويه، هما: الانتشار **Diffusion** والخلط **Confusion**. نعرض فيما تعريفاً لهذين الأسلوبين ونوضح بالشكل مكونات مشفرات الجداء:

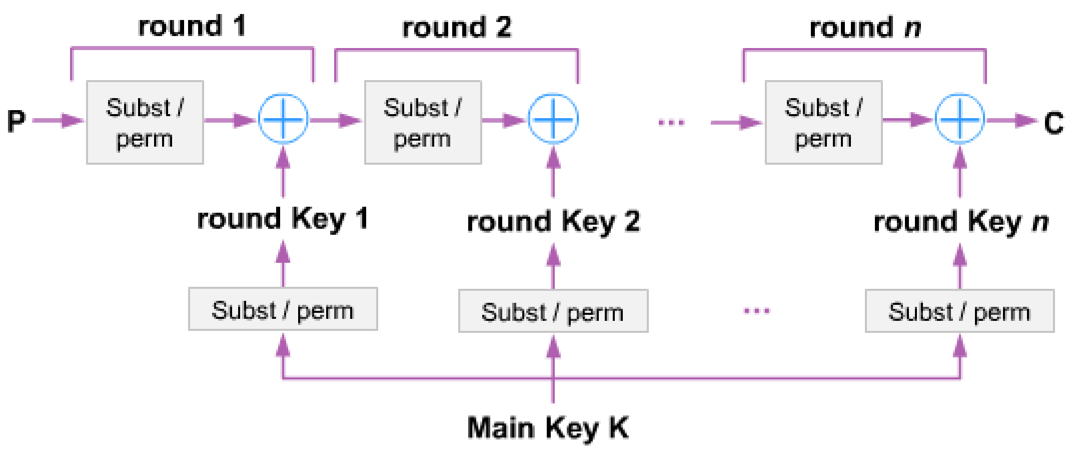
- **الانتشار Diffusion**: تفيد هذه العملية في إخفاء العلاقة بين النص غير المشفر والنص المشفر حيث يمكن أن يؤدي تغيير خانة واحدة (بت) ضمن النص غير المشفر إلى تغيير عدة خانات ضمن النص المشفر.
- **الخلط Confusion**: تفيد هذه العملية في إخفاء العلاقة بين النص المشفر ومفتاح التشفير حيث يمكن أن يؤدي تغيير خانة واحدة (بت) ضمن مفتاح التشفير إلى تغيير عدة حروف ضمن النص المشفر.
- يبين الإيضاح أماكن استخدام أسلوبي الانتشار والخلط.



- **الجولات Rounds**: يمكن أن تتكرر مشفرات الجداء المكونة من مكونات صناديق التدوير وصناديق الاستبدال المزودة بإمكانيات الانتشار والخلط أكثر من مرة من وراء بعضها البعض. نسمي كل تكرار بالجولة **Round**. ويستخدم المشفر الكتلي عندها مجدول للمفاتيح **Key schedule** أو مولد لها **Key generator** لإنشاء مفاتيح مختلفة لكل جولة.

يجري ضمن مشفر مؤلف من **N** جولة تشفر النص غير المشفر **N** مرة قبل توليد النص

المشفر النهائي. نسمي النص المولد بين الجولات بالنص الوسيط **Middle text**. يوضح الشكل التالي بنية مشفر جداء مؤلف من عدة جولات.



صفوف مشفرات الجداء Classes of Product Ciphers

كافة المشفرات الكتلية الحديثة هي مشفرات جداء. ولكن يمكن تصنيف هذه المشفرات عموماً ضمن صفين:

- تتميز مشفرات الصف الأول باستخدام ضمن بنيتها مكونات قابلة للعكس وأخرى غير قابلة للعكس وتسمى بالمشفرات **Feistel Ciphers** وتعتبر المشفر **DES** أحد هذه المشفرات المعروفة والمستخدمة بكثرة.
- أما مشفرات الصف الثاني وتسمى بالمشفرات **Non-Feistel Ciphers**، فهي تعتمد على استخدام مكونات قابلة للعكس فقط ضمن بنيتها ويعتبر المشفر **AES** مثالاً حقيقياً لهذا الصف من المشفرات.

2. مشفرات دفق المعلومات الحديثة Modern Stream Ciphers

يجري التشفير مع هذه المشفرات على مستوى الخانات (بتات) وفقاً لما يلي:

$$P = P_1 P_2 P_3 \cdots$$
$$C = C_1 C_2 C_3 \cdots$$
$$K = (k_1, k_2, k_3, \cdots)$$
$$C_1 = E_{k1}(P_1)$$
$$C_2 = E_{k2}(P_2)$$
$$C_3 = E_{k3}(P_3) \cdots$$

يتوفر لدينا نوعين من مشفرات دفق المعلومات الحديثة، هما:

- **المشفرات المتزامنة** **Synchronous Stream Ciphers**: وفيها يكون دفق معلومات المفتاح مستقلاً عن دفق معلومات النص غير المشفر أو المشفر. يجري توليد دفق معلومات المفتاح واستخدامه دون وجود أية علاقة بين خاناته وخانات النص غير المشفر أو المشفر.
- **المشفرات غير المتزامنة** **Nonsynchronous Stream Ciphers**: وفيها يكون دفق معلومات المفتاح مرتبطاً بالنص غير المشفر أو المشفر.

الفصل الثالث

معيار تشفير المعطيات

Data Encryption Standard (DES)

الأهداف Objectives

يهدف هذا الفصل إلى التعريف بما يلي:

- عرض نبذة تاريخية عن التقنية DES.
- تعريف بالبنية الأساسية لـ DES ووصف مكوناتها.
- وصف طريقة توليد المفاتيح لجولات التشفير في DES.

1. مقدمة Introduction

المعيار DES هو:

مشفر كتلي متناظر المفتاح. جرى نشره من قبل المعهد الوطني للمعايير والتقانة **National Institute of Standards and Technology (NIST)**.

نبذة تاريخية History:

في عام **1973** طلب المعهد **NIST** تقديم مقترحات لبناء نظام وطني لتشفير متناظر المفتاح. تقدمت شركة **IBM** بمقترحها الذي قبل فيما بعد ونشر تحت المسمى **DES** في عام **1975**.

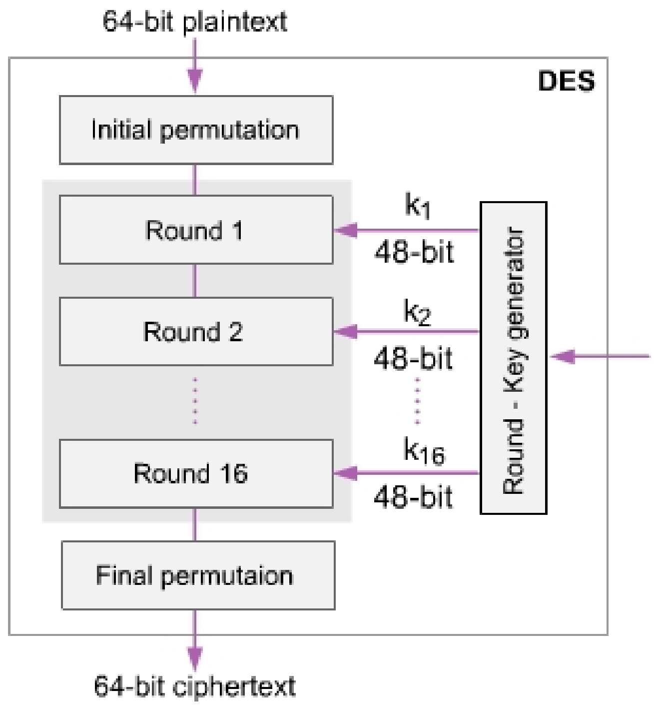
بعد النشر، تعرض المشفر إلى الكثير من النقد وذلك لسببين. تمثل الأول في صغر طول المفتاح (فقط **56** بت)، مما يجعله عرضة للكسر بسهولة. أما السبب الثاني، فيعود إلى إخفاء بنية بعض المكونات الداخلية له (المكونات المرتبطة بصناديق الاستبدال) مما أثار مخاوف من وجود أبواب خلفية متروكة من المصممين للاطلاع والتجسس على ما يقوم به المشفر.

جرى فيما بعد إصدار نسخة مطورة عن **DES** تحت المسمى **Triple DES** (لأنه يقوم بتكرار عمل **DES** ثلاث مرات). ومن المتوقع أن يحل المشفر **AES** محل المشفر **DES** مستقبلاً.

2. بنية المعيار DES Structure DES

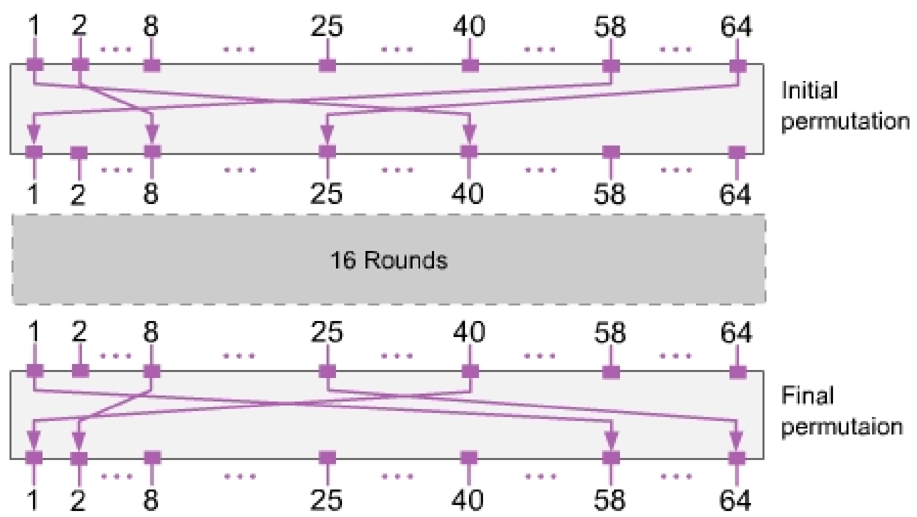
تجري عملية التشفير بواسطة صندوقي تدوير **P-Boxes** و**16** جولة تشفير من النوع **Feistel**.
تستخدم كل جولة مفتاحاً مختلفاً بطول **48** خانة. يجري توليد هذه المفاتيح بواسطة مولد لها من

مفتاح التشفير الرئيسي باستخدام خوارزمية سنتحدث عنها لاحقاً. يبين الشكل التالي عناصر المشفر **DES**:



التدوير الأولي والنهائي Initial and Final Permutations

يعرض الشكل التالي بنية صندوقي التدوير الأولي والنهائي المستخدمين مع المشفر **DES**:



يأخذ كلا صندوقي التدوير سلسلة خانات من **64** بت كدخل وتعطي كخرج سلسلة أخرى بعد تدوير الخانات من **64** بت أيضاً. عمليات التدوير هذه هي عمليات تشفير بدون مفتاح وهي من

النوع المباشر. كما أن هاتين العمليتين متعاكستين، حيث نلاحظ أن البت رقم **58** في دخل مرحلة التدوير الأولي ستصبح البت رقم واحد من خرج هذه المرحلة. وبنفس الطريقة تصبح البت الأولى من دخل مرحلة التدوير النهائي البت رقم **58** ضمن خرج هذه المرحلة.

بكلمات أخرى، يمكننا القول أنه إذا لم يكن لدينا الجولات الست عشر بين مرحلتي التدوير الأولي والنهائي فإن الخانة رقم **58** من دخل مرحلة التدوير الأولي ستخرج من نفس الموضع رقم **58** من خرج مرحلة التدوير النهائي. يبين الجدول التالي قواعد التدوير لعمل كلا صندوقي التدوير:

(a) Initial permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

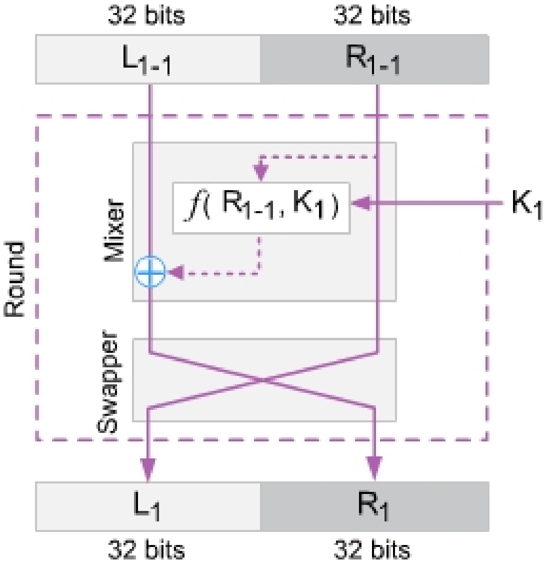
(b) Inverse Initial permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

يجب ملاحظة أن مرحلتي التدوير اللتين تحدثنا عنهما فيما سبق ليس لهما أي دلالة بالنسبة للمشفر **DES**.

الجولات Rounds

تتكون كل جولة من مشفر **Feistel** كما يظهر بالإيضاح التالي:



تأخذ كل جولة الجزئين: L_{I-1} و R_{I-1} من الجولة السابقة لتقوم بإنشاء الجزئين: L_I و R_I اللذين سيصبحان دخل للجولة التالية. تتضمن كل جولة عنصري التشفير: الخلاط **Mixer** والمبدل **Swapper**. وكلا هذين العنصرين قابل للعكس. من الواضح أن المبدل قابل للعكس بسبب طبيعته

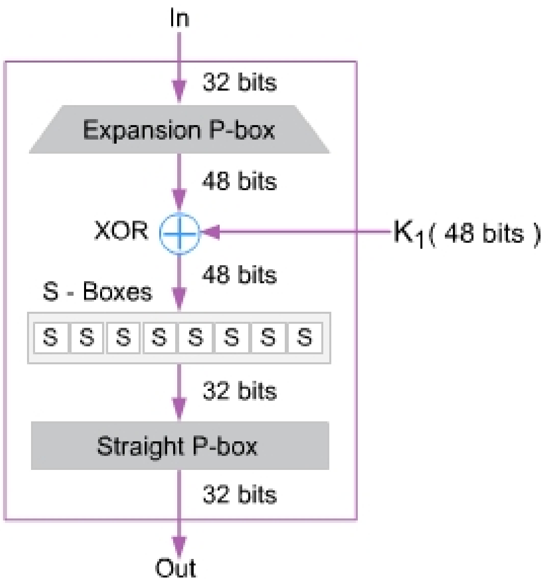
التي تبدل بين الجزء الأيسر مع الجزء الأيمن. أما الخلاط فهو قابل للعكس أيضاً لأنه يعتمد في عمله على العملية **XOR**. جرى تجميع كافة العناصر غير القابلة للعكس ضمن التابع $f(R_{I-1}, K_I)$.

تابع المشفر DES Function

يعتبر تابع المشفر **DES** هو قلب المشفر. يقوم هذا التابع بتطبيق المفتاح من الطول **48** بت على الجزء R_{I-1} المكون من **32** بت. يتألف هذا التابع من أربعة مقاطع، هي:

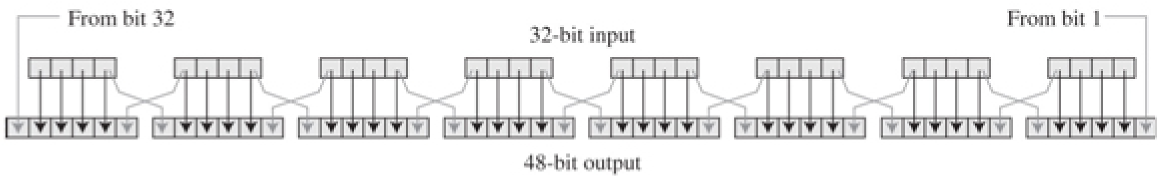
1. صندوق تدوير موسع.
2. المبيض الذي يقوم بإضافة المفتاح.
3. مجموعة من صناديق الاستبدال.
4. صندوق تدوير مباشر.

كما هو مبين في الإيضاح التالي:



صندوق تدوير موسع Expansion P-box

على اعتبار أن طول الجزء R_{I-1} يساوي **32** بت وأن طول المفتاح هو **48** بت فإنه يجب زيادة **16** بت على الجزء المعالج كما هو مبين في الشكل التالي، حيث يجري تقسيم بتات الجزء R_{I-1} إلى **8** أجزاء مؤلفة من **4** بت. ثم يوسع كل جزء ليصبح مؤلفاً من **6** بت بدلاً من **4**



يتم التوسيع وفقاً لقواعد محددة مسبقاً مبينة في الجدول التالي:

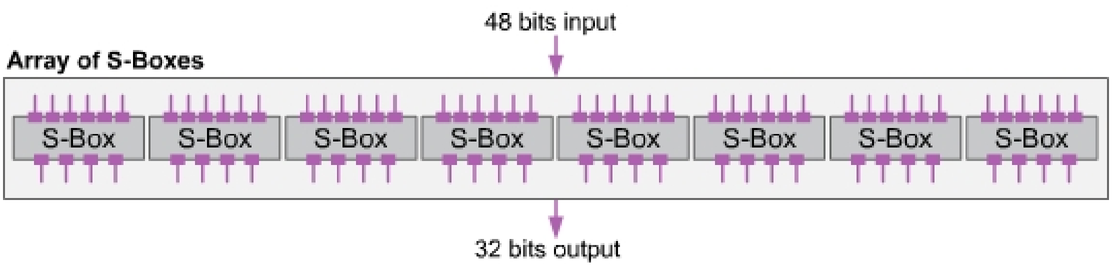
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

المبيض Whitener (XOR)

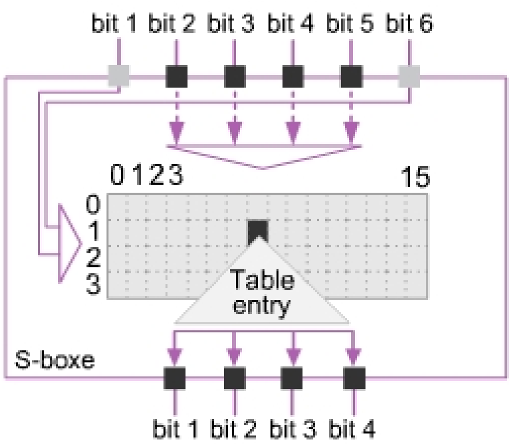
يعتمد المشفر **DES** الذي يقوم بتطبيق التابع **XOR** على الجزء الأيمن الموسع من الدخل ومفتاح الجولة الذي يجري استخدامه فقط مع هذا التابع فقط.

مجموعة من صناديق الاستبدال S-Boxes

يهدف استخدام هذه الصناديق إلى تنفيذ عملية الخلط الحقيقية. يعتمد المشفر **DES** على استخدام **8** صناديق استبدال، يأخذ كل منها **6** خانات كدخل ويعطي **4** خانات كخرج، كما هو مبين في الشكل التالي:



تجري عملية الاستبدال اعتماداً على جدول مؤلف من **4** أسطر و**16** عمود، كما هو مبين في الشكل التالي:



وعلى اعتبار أن يوجد لدينا **8** صناديق استبدال فإنه لدينا ثمانية جداول، حيث يجري يحدد رقم السطر المستهدف من خلال القيمة العشرية الناتجة عن تجميع قيمتي الخانة الأولى مع السادسة من الدخل، وتحدد قيمة الخرج من خلال القيمة العشرية الناتجة عن تجميع قيم الخانات من **2** إلى **5**. نعرض فيما يلي محتوى الجداول الثمانية المستخدمة ضمن صناديق الاستبدال.

s_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
s_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
s_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
s_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
s_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
s_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
s_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
s_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

صندوق التدوير المباشر Straight Permutaion

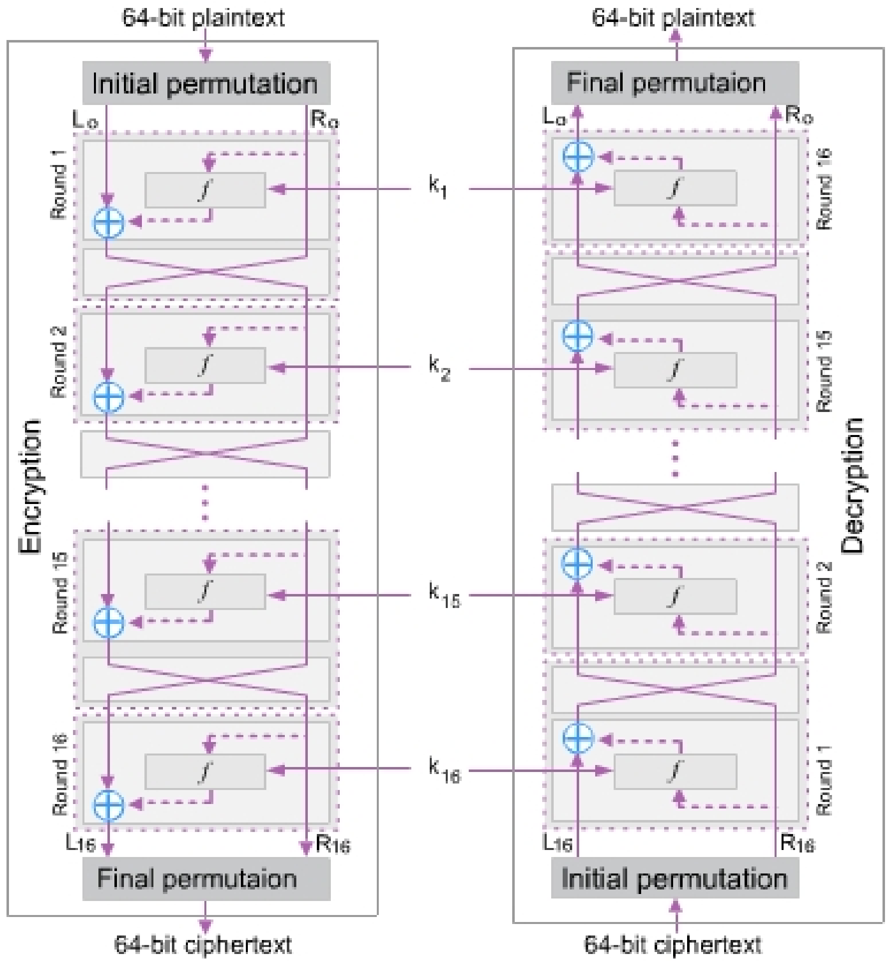
يهدف استخدام هذه الصناديق إلى تنفيذ عملية التدوير وفقاً للجدول التالي:

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

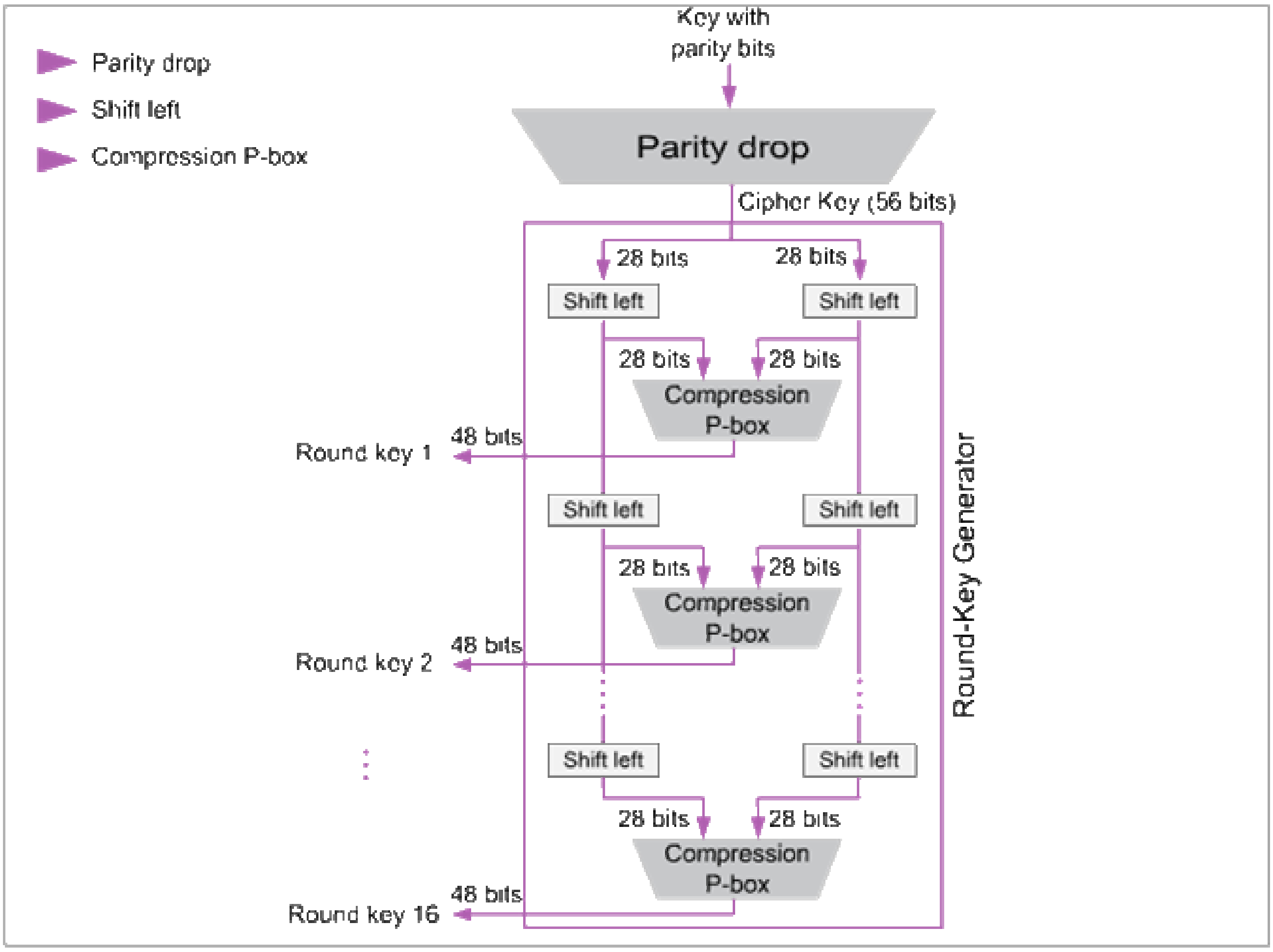
التشفير وعكس التشفير Cipher and Reverse Cipher

باستخدام الخلاطات والمبدلات، يمكن تنفيذ عملية التشفير وفك التشفير التي تتضمن كل منها 16 جولة. لتحقيق ذلك، نجعل الجولة الأخيرة رقم 16 مختلفة عن بقية الجولات وتحتوي على خلاط دون مبدل كما هو مبين في الجدول التالي:



توليد المفاتيح Key Generation

يهدف هذا الجزء إلى إنشاء 16 مفتاح بطول 48 بت باستخدام مفتاح رئيسي بطول 56 بت.
يجري عادة التعامل مع المفتاح الرئيسي كما لو أنه مؤلف من 64 بت (56 بت + 8 بت إضافية كزوجية Parity bits) ولكن يجري تجاهل خانات الزوجية قبل الدخول فعلياً بعملية توليد المفاتيح، كما هو مبين في الشكل التالي:



تجاهل خانات الزوجية Parity Drop:

يقوم جزء تجاهل خانات الزوجية (وهي الخانات 8، 16، 24، 32، ....، 64) من الخانات 64 المكونة للمفتاح الرئيسي. يجري بعدها تدوير بقية الخانات كما هو مبين في الجدول التالي:

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

الإزاحة إلى اليسار Shift Left:

بعد عملية التدوير المباشرة، يجري تقسيم المفتاح إلى جزئين بطول 28 خانة. يجري بعدها إزاحة كل جزر إلى اليسار (إزاحة دوارنية) خانة أو خانتين. في الجولات 1 و2 و9 و6 تنفذ عملية إزاحة لخانة واحدة أما في بقية الجولات، فتنفذ إزاحة من خانتين يجمع بعدها كلا الجزئين ليشكلا جزءاً واحداً من 56 خانة. يبين الجدول التالي عدد خانات الإزاحة في كل جولة:

Round number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

صناديق التدوير المضغوطة Compression Permutation:

تهدف هذه الصناديق إلى تغيير 58 خانة إلى 48 خانة والتي يجري استخدامها كمفتاح. يبين الجدول التالي كيفية تنفيذ عملية التدوير المضغوطة:

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

3. تحليل المعيار DES Analysis DES

تعرض المعيار DES خلال السنوات السابقة إلى الكثير من النقد. ولقد خضع إلى الكثير من الاختبارات لتحديد فيما إذا كان محققاً للخصائص المرجوة منه.

تبين أن هذا المعيار مقاوم لعدة أشكال من عمليات التحليل. إلا أن صغر حجم مفتاحه جعله عرضة إلى هجومات البحث المكثف التي يمكن أن تؤدي إلى كسر التشفير الذي يقوم به. من بين الحلول الممكنة لهذه المشكلة، يمكن التفكير بما يلي:

مضاعفة حجم المفتاح وهذا الأمر غير ممكن مع المعيار **DES**، تطبيق المشفر **DES** مرتين على نفس المعطيات (إلا أن هذا الحل معرض لخطر الهجوم "قابلني في الوسط" **Meet in middle**)، وتطبيق المشفر ثلاث مرات على نفس المعطيات (وهو المعيار الحالي المسمى بالمعيار **DES** الثلاثي **Triple DES**).

سنتحدث فيما يلي عن هذه الأفكار والمواضيع:

1. أثر الانهيار الثلجي.
2. الهجمات التحليلية.
3. هجمات البحث المكثف.
4. تكرار المشفر.

أثر الانهيار الثلجي Avalanche Effect

من بين الخصائص التي يجب أن يتمتع بها المشفر لدينا ما يسمى بأثر الانهيار الثلجي **Avalanche Effect** وهو يحدد مقدار الأثر الذي تتركه التعديلات المنجزة ضمن المعيار على ملف الدخل (النص غير المشفر) وأثرها على النص المشفر.

يبين الجدول المجاور أثر استخدام الدخل لتوليد مفاتيح الجولات ليحدد عدد البتات المعدلة ضمن النص غير المشفر والتي أدت إلى تغيير بتات مقابلة ضمن النص المشفر. كما يبين هذا الجدول أثر استخدام عمليات الإزاحة والتدوير أثناء توليد المفاتيح على عدد البتات المتأثرة من جراء ذلك ضمن النص المشفر ضمن كل جولة من الجولات.

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

نلاحظ من الجدول السابق الأثر المحدود للتغيرات التي تم انجازها ضمن كل جولة من جولات المعيار **DES**.

الهجمات التحليلية Cryptoanalysis Attacks

أدت تحليل نتائج الهجومات التحليلية إلى ظهور النتائج التالية:

- بالنسبة إلى التحليلات التفاضلية **Differential Cryptoanalysis**، تبين أن وجود **16** جولة ضمن المعيار هو لمقاومة هذا النوع من التحليلات. فقد تبين أننا نحتاج إلى 2^{47} نص غير مشفر معروف لكسر المعيار **DES**.
- أما بالنسبة للتحليلات الخطية **Linear Cryptoanalysis**، تبين أن المعيار **DES** غير مصمم لمثل هذه الهجومات (التي ظهرت بعد إطلاق المعيار) ولكن ظهر أن صناديق الاستبدال **S-Boxes** الخاصة بالمعيار غير خطية، حيث نحتاج إلى 2^{43} نص غير مشفر معروف لكسر المعيار **DES**.

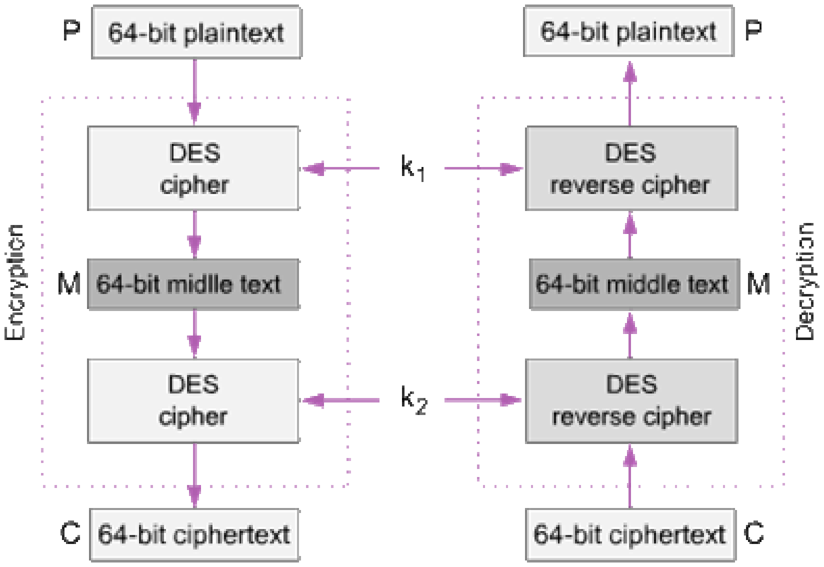
هجمات البحث المكثف Exhaustive Search Attacks

لا تعتبر المفاتيح بطول **56** بت آمنة من الناحية الحسابية. يمكن أن تؤدي هجومات المعالجة التفرعية إلى كسرالمعيار **DES**:

- أدى استخدام حاسوب مزود بمليون دارة **Chips** عام **1998** إلى إيجاد المفتاح خلال **112** ساعة.
- أدى استخدام شبكة من **3500** حاسوب (عام **1977**) إلى إيجاد المفتاح خلال **120** يوم.

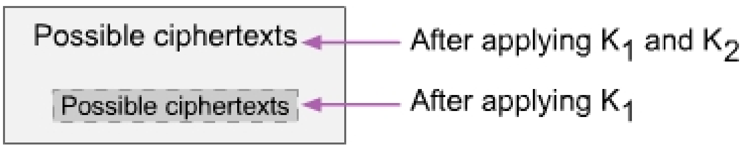
تكرار المشفر DES Multiple DES

لا يعتبر المفتاح بطول **56** بت خياراً آمناً ولا يوجد بالنسبة للمعيار **DES** لتكبير المفتاح. وبالتالي فإن الحل الوحيد يتمثل في تكرار المشفر **DES** أكثر من مرة **Multiple DES** مع استخدام مفتاح مختلف لكل مرحلة ويكون خرج كل مرحلة دخل للمرحلة التالية، كما هو مبين في الشكل التالي:



يوضح هذا الشكل كيفية بناء مشفر **DES** مكون من مرحلتين وبمفتاحين مختلفين.

يؤدي وجود المراحل المتكررة مع استخدام مفاتيح مختلفة إلى زيادة عدد الاحتمالات المختلفة للنص المشفر بشكل كبير جداً كما هو مبين في الشكل التالي الذي يبين عدد احتمالات النص المشفر في حالة مرحلة واحدة وفي حالة مرحلتين:



يتوفر لدينا في هذه الحالة: **!**(2^{64}) إمكانية مطابقة بين 2^{64} كتلة دخل ممكنة مع 2^{64} كتلة خرج ممكنة. كما يوجد لدينا 2^{56} مفتاح ممكن وهذا يعتبر جزء بسيط من الحجم السابق.

تبين الأرقام السابقة أنه من الصعب جداً إيجاد مفتاح K_3 بحيث نحصل على ما يلي:

$$E(E(P, K_1), K_2) = E(P, K_3)$$

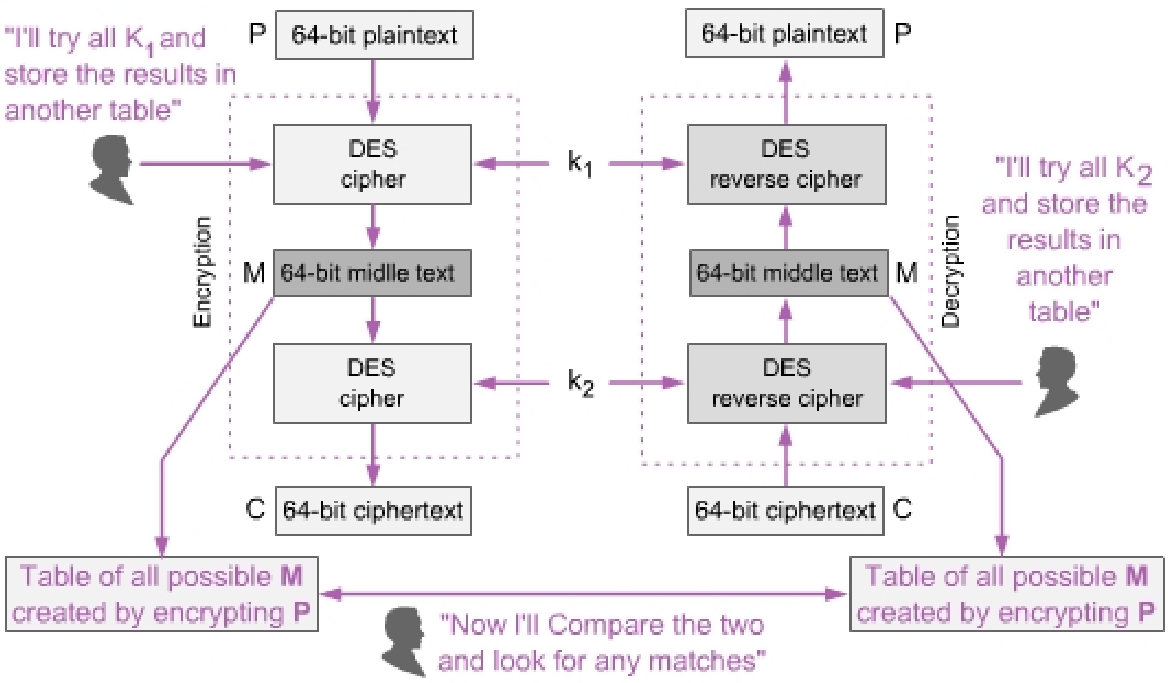
من المفترض نظرياً أن يشكل المشفر من مرحلتين **DES** حلاً مناسباً لأنه من أجل كسر التشفير نحتاج إلى تجريب كل التركيبات الممكنة للمفتاحين K_1 و K_2 والتي يبلغ عددها $2^{56} \times 2^{56} = 2^{112}$ تركيب ممكن.

إلا أن هذا المشفر معرض لخطر الهجوم "قابلني في الوسط" **Meet in middle** الذي يمكن تلخيص طريقة عمله كما يلي:

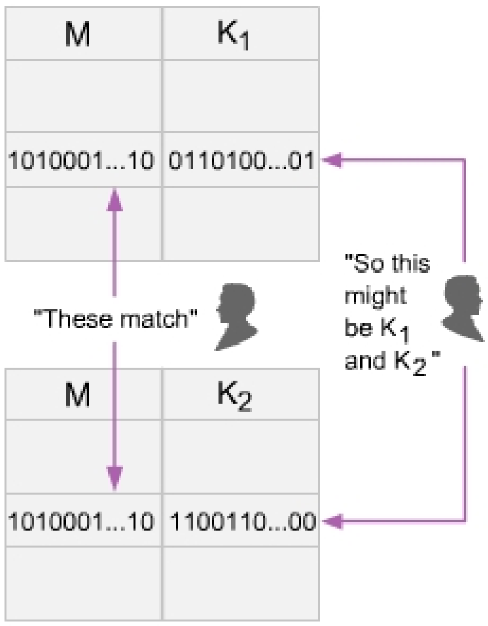
- نأخذ نص معروف غير مشفر **P** ونحسب النص المشفر الناتج عنه **C**.
- نعمل باتجاه الأمام ونحسب كافة الأشكال الممكنة للمفتاح K_1.

- نعمل باتجاه الخلف ونحسب كافة الأشكال الممكنة للمفتاح K_2.
- نخزن النتائج ونبحث عن التطابق بين النتائج.

يبين الشكل التالي طريقة العمل هذه:

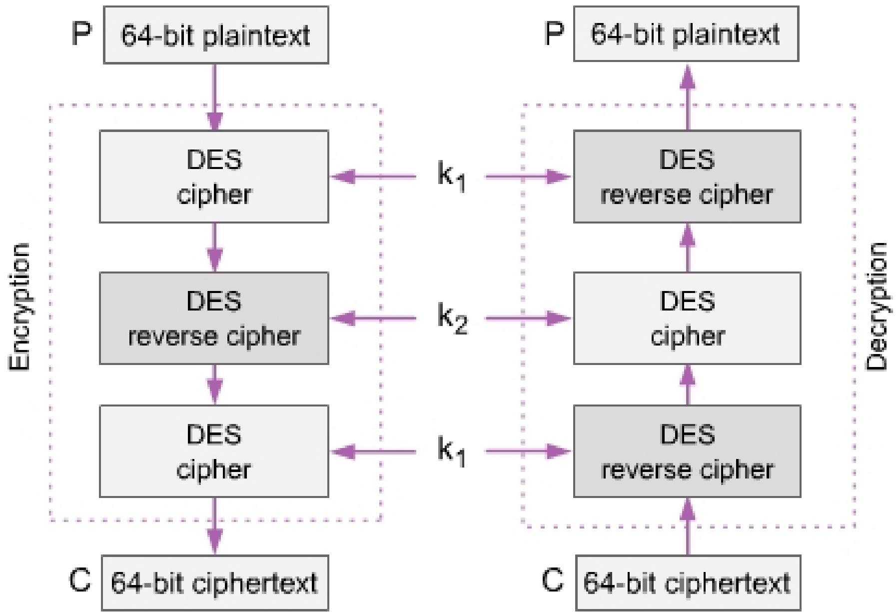


بعد تخزين النتائج نحصل على جدولين للمطابقة بينهما وتحديد المفتاحين:



نحتاج إلى 2^{56} تشغيل لتوليد كل جدول وإلى تنفيذ 2^{56} x 56 عملية مقارنة لإيجاد التطابق وتحديد المفتاحين المطلوبين. تؤكد النتائج السابقة أن **DES** المضاعف والمؤلف من مرحلتين غير آمن حسابياً.

لتجاوز المشكلة السابقة، نحتاج إلى مرحلة **DES** إضافية كما هو مبين في الشكل:



وهو الحل المستخدم حالياً في الكثير من الأماكن وخاصة في عالم البنوك.

الفصل الرابع

معيار تشفير المعطيات المتقدم

(Advanced Encryption Standard AES)

الأهداف Objectives

يهدف هذا الفصل إلى التعريف بما يلي:

- عرض نبذة تاريخية عن التقنية **AES**.
- تعريف بالبنية الأساسية لـ **AES** وصف مكوناتها.
- وصف عملية توسيع المفاتيح **Key expansion process**.

1. مقدمة Introduction

التقنية **AES** هي مشفر كتلي متناظر. نشرت هذه التقنية من قبل المعهد الوطني للمعايير والتقانة **National Institute of Standards and Technology (NIST)** في عام **2001**.

نبذة تاريخية History:

في عام **1997** باشر المعهد **NIST** بالبحث عن بديل لتقنية **DES** تحت المسمى "المعيار **AES**". تضمنت متطلبات تصميم المعيار النظام التعامل مع كتل من حجوم مساوية لـ **128** بت، ومع مفاتيح من ثلاثة حجوم مختلفة وهي: **128** و**192** و**256** بت. كما تضمنت المتطلبات أن تكون خوارزمية المعيار مفتوحة ومعروفة من قبل العامة.

جرى في عام **1999** اختيار بعض الخوارزميات على أنها محققة تماماً للمتطلبات. نذكر من بينها:

- خوارزمية **Rijndael**.
- خوارزمية **Serpent**.
- خوارزمية **Twofisk**.

وغيرها من الخوارزميات التي ما تزال مستخدمة حتى الآن.

في عام **2001** قام المعهد **NIST** بنشر خوارزمية **Rijndael** على أنها المعيار **AES**.

متطلبات تصميم المعيار Criteria

تصب متطلبات تصميم المعيار التي اعتمدت والتي بناء عليها جرى اختيار خوارزمية **Rijndael** ضمن ثلاثة فئات، هي:

- **الأمن Security:** على اعتبار أن المعيار **AES** معتمد على التعامل مع مفاتيح بطول **128**

بت فهو محقق لمتطلبات الأمن حتى يومنا هذا. كما أن القدرة على التعامل مع مفاتيح بطول 192 و256 بت ستضيف إمكانيات أمنية أخرى للمستقبل.

- **الكلفة Cost**: جرى تصميم النظام بشكل فعال ومنسجم مع البنى الحاسوبية والبرمجية الموفرة حالياً.
- **التحقيق Implementation**: يمكن تحقيق هذا المعيار باستخدام أية منصة عمل.

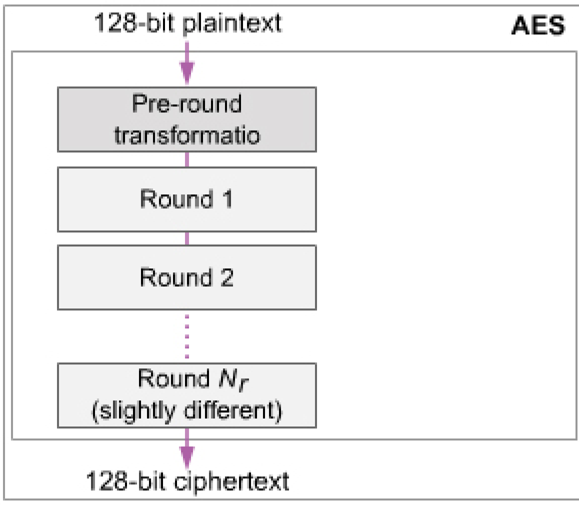
2. بنية المعيار AES Structure AES

AES هو:

مشفر كتلي من النوع **Non-Feistel**.

يتمتع هذا المعيار بالقدرة على التعامل مع كتل بحجم **128** بت ويقوم باستخدام **10** أو **12** أو **14** جولة وذلك تبعاً لطول المفتاح المستخدم والذي يمكن أن يكون **128** أو **192** أو **256** بت على التوالي. يوضح الشكل التالي البنية العامة لخوارزمية تشفير هذا المعيار (التي تسمى بالمشفر). تشبه خوارزمية فك التشفير في بنيتها بنية المشفر ولكن يجري معها تطبيق المفاتيح بطريقة معكوسة.

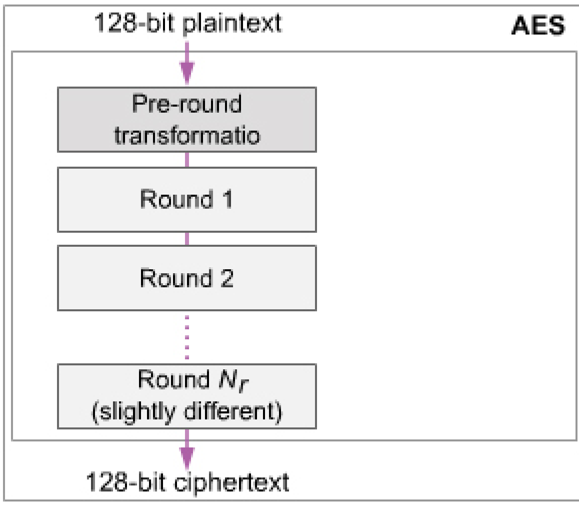
تحدد القيمة N_r في الشكل عدد جولات المشفر .
يبين الإيضاح العلاقة بين عدد الجولات وطول المفتاح.



N_r	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

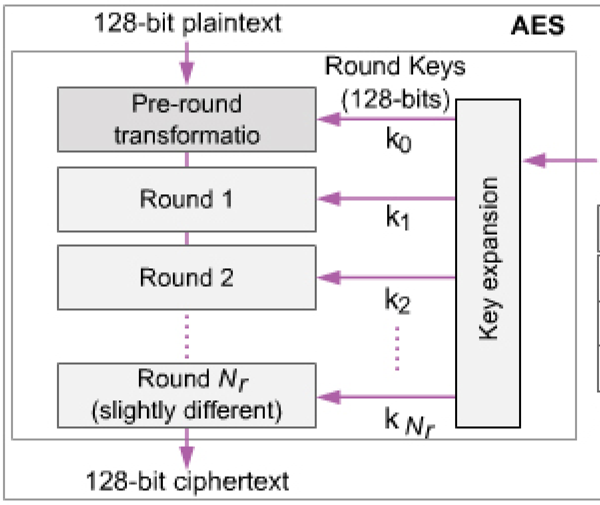
يبين أنه يتوفر لنا ثلاث نسخ من مشفر **AES** و ذلك حسب طول المفتاح المستخدم. نطلق على هذه النسخ المسميات التالية: **AES-128** و**AES-192** و **AES-256** لكن يبقى طول المفاتيح تشفير الجولات مع النسخ الثلاثة بطول **128** بت مماثل لطول كتل معطيات النص المشفر وغير المشفر التي يجري التعامل معها.



N_r	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

يجري توليد مفاتيح تشفير الجولات بعدد يزيد بواحد عن عدد هذه الجولات ونسمي هذه المفاتيح ما يلي: $k_0, k_1, k_2, \ldots, k_N$.



N_r	Key size
10	128
12	192
14	256

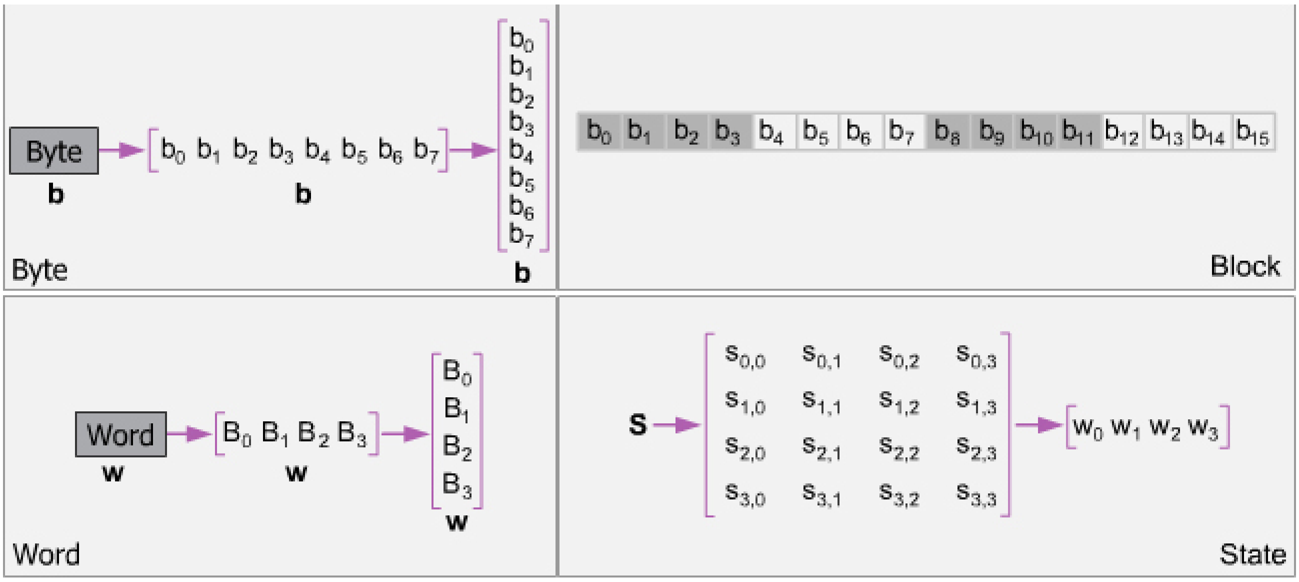
Relationship between number of rounds and cipher key size

ملاحظة هامة

يتعامل المعيار **AES** مع خمسة وحدات من المعطيات، هي:

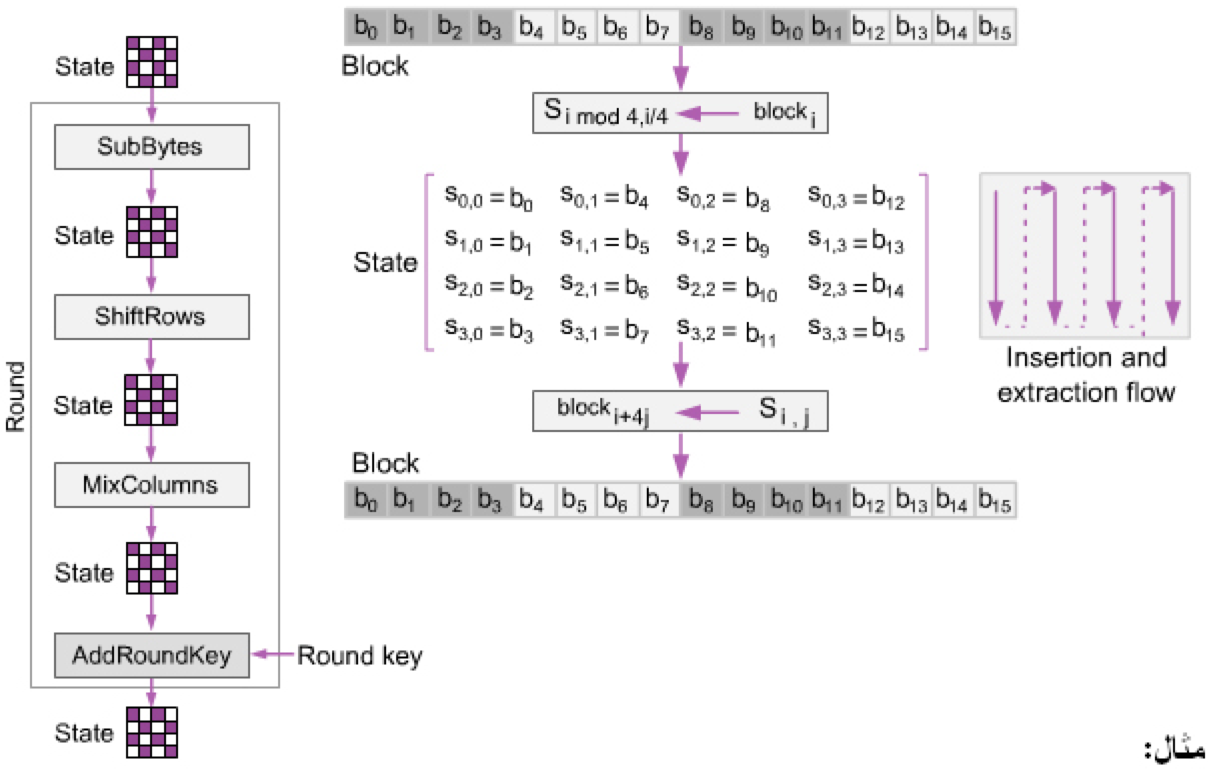
البت Bit والبايت Byte والكلمة Word والكتلة Block والحالة State.

يبين الإيضاح التالي حجوم هذه المعطيات، وطريقة تمثيلها على شكل مصفوفات سطرية أو أعمدة:



يقوم المعيار **AES** بالتعامل مع عدة جولات وتتألف كل جولة من عدة مراحل. يجري تحويل المعطيات من مرحلة إلى أخرى. نسمي المعطيات في بداية التشفير ونهايته بكتل المعطيات. أما في بداية ونهاية كل مرحلة فنسميها بالحالة **State**. تشبه الحالات الكتل وهي مؤلفة من **16** بايت، ولكن يجري تمثيلها على شكل مصفوفات **4x4** **(S)** حيث يعبر عن كل عنصر بالشكل $s_{r,c}$. يمكن أحياناً التعبير عن الحالة على شكل مصفوفة **1x4** من الكلمات (يعتبر ذلك منطقياً إذا نظرنا إلى كل عمود من أعمدة المصفوفة الأصلية على أنه كلمة).

في بداية عملية التشفير، يجري إضافة بايتات المعطيات المكونة لكتل المعطيات ضمن مصفوفة الحالة عمود فعمود وضمن كل عمود من الأعلى إلى الأسفل. في نهاية التشفير، يجري استخراج البايتات من الحالة بنفس الطريقة، كما هو مبين في الشكل التالي:



مثال:

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19

State

00	12	0C	08
04	04	00	23
12	12	13	19
14	00	11	19

بنية الجولات Structure of Each Round

يبين الشكل التالي بنية كل جولة من جولات التشفير المكونة للمعيار **AES**. تقوم كل جولة (ما عدا الجولة الأخيرة) باستخدام أربع تحويلات **Transformation** قابلة للعكس. أما الجولة الأخيرة فتستخدم ثلاث تحويلات فقط.

يوضح المجاور أن كل تحويل يبدأ بحالة ثم ينشئ حالة جديدة. قبل الدخول إلى الجولات، يجري تنفيذ تحويل إبتدائي **transformation Pre-round**، وهو مكون من تحويل واحد هو تحويل إضافة مفتاح الجولة **AddRoundKey**. أما الجولة الأخيرة، فتضم ثلاث تحويلات فقط ويكون تحويل خلط الأعمدة **MixColumns** مفقوداً ضمنها.

عند فك التشفير، يجري التعامل مع تحويلات معاكسة للتحويلات السابقة، وهي:

InvSubByte و**InvShiftRows** و **InvMixColumns** و **AddRoundKey** (الذي يتمتع بكونه قابل للعكس ذاتياً).

3. البنى الرياضية الخاصة بـ AES Mathematics AES

نحتاج لفهم طريقة عمل التحويلات المرتبط بنية المعيار **AES** إلى التذكير ببعض البنى الجبرية التي يستند إليها تعريف التحويلات رياضياً وهي:

1. باقي قسمة ضرب عددين على عدد آخر **Modular Mutliplication**.
2. عكوس باقي قسمة ضرب عددين على عدد آخر **Modular Mutliplication Inverses**.
3. حقول غالواس **Galois Fields**.
4. عكوس قيم حقول غالواس **Galois Field Inverses**.

تساعد هذه البنى على توفير خصائص هامة للتحويلات والتي نذكر منها:

- عدم الخطية **Non-linearity** في عمل هذه التحويلات.
- يمكن أن تعمل كما نريد عند التعامل مع المفاتيح الطويلة.
- يجب أن تظهر عمليات المطابقة وتحديد الخيار الممكن كما لو أنها عشوائية تماماً.

باقي قسمة ضرب عددين على عدد آخر Modular Mutliplication

نحسب باقي قسمة العددين **a** و**b** على العدد **m** وفقاً للعلاقة التالية: $(a*b) \bmod m$.

مثال1: يمثل الجدول التالي نتيجة ضرب عددين مع قسمة النتيجة على العدد **7**:

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

نقول أن العدد **b** يمثل عكس للعدد **a mod m** إذا تحقق ما يلي: $(a*b) \bmod m = 1$ أي أن $b = a^{-1} \bmod m$.

نحسب باقي قسمة العددين a وb على العدد m وفقاً للعلاقة التالية: $(a*b) \bmod m$.

مثال2: $5 = 3^{-1} \bmod 7$ على اعتبار أن $3 \times 5 = 15 \bmod 7 = 1$ كما مبين فيما يلي:

a	a^{-1}
0	None
1	1
2	4
3	5
4	2
5	3
6	6

×	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

عكوس باقي قسمة ضرب عددين على عدد آخر Modular Mutliplication Inverses

لا يمكن تنفيذ هذه العملية إلا باستخدام قيمة أولية لـ m وإلا سينتج لدينا أعداد ليس لها عكوس كما هو مبين في الشكل التالي(m=8 وهي ليست قيمة أولية):

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	1	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	1	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	1

a	a^{-1}
0	none
1	1
2	none
3	3
4	none
5	5
6	none
7	7

الهدف من حساب العكوس هو أن نستخدم نفس الفكرة مع قيمة لـ $m = 2^n$ (حيث أن m هي الحجم الافتراضي لكتلة معطيات).

حقول غالواس Galois Fields

نحتاج هذه البنية من أجل التعبير عن تمثيل البايت لتحويلها إلى كثير حدود. كما نستخدمها من أجل حساب العكوس كما سنبين فيما بعد. تستخدم الحقول مع قيمة $m = 2^n$ من أجل بناء

صناديق الاستبدال للمعيار **AES**، وللمطابقة بين **256** بايت ممكنة للدخل مع **256** بايت ممكنة للخرج.

حقول غالواس رياضياً:

الخطوة الأولى:

نقوم أولاً بتمثيل الأعداد الثنائية على شكل كثيرات للحدود، كما هو مبين في الجدول التالي من أجل الحقل $GF(2^3)$:

000	$0x^2 + 0x + 0$	0
001	$0x^2 + 0x + 1$	1
010	$0x^2 + 1x + 0$	x
011	$0x^2 + 1x + 1$	x + 1
100	$1x^2 + 0x + 0$	x^2
101	$1x^2 + 0x + 1$	$x^2 + 1$
110	$1x^2 + 1x + 0$	$x^2 + x$
111	$1x^2 + 1x + 1$	$x^2 + x + 1$

نلاحظ أن أمثال الحدود **coefficients** هي ثنائية القيمة (إما **0** أو **1**). وتجري عملية الجمع والطرح ضمن المجال **mod 2**.

مثال:

$$\begin{array}{r} x^2 \\ -\ (x+1) \\ \hline x^2 - x - 1 \\ = x^2 + x + 1 \end{array} \qquad \begin{array}{r} x^2 + x + 1 \\ +\ x + 1 \\ \hline x^2 + 2x + 2 \\ = x^2 + 0x + 0 \\ = x^2 \end{array}$$

على اعتبار أن $-1 \bmod 2 = 1$ وأن $2 \bmod 2 = 0$.

الخطوة الثانية:

نبحث عن كثير الحدود الأولي P_n من الدرجة **n**، وهو كثير الحدود الذي ليس هو من مضاعفات أي كثيري حدود آخرين (ما عدا **1** ونفسه).

مثال: بالنسبة للحقل $GF(2^3)$ لدينا كثير الحدود $P_3 = x^3 + x + 1$.

مثال آخر: بالنسبة للحقـل $GF(2^8)$ المستخدم مع المعيار **AES** لدينا كثير الحدود $P_8 = x^8 + x^4 + x^3 + x + 1$.

الخطوة الثالثة:

نحسب جدول الضرب لكافة الثنائيات لحساب المعادلة التالية $P_i \times P_j \bmod P_n$. نحتاج إلى حساب الباقي بواسطة التابع **mod** عندما يكون لدينا درجة كثير الحدود الناتج عن $P_i \times P_j$ أكبر أو تساوي **n**. يمكن تبسيط عملية الحساب بأن نقوم بحساب المعادلة التالية $P_i \times P_j - x^{k-n} \times P_n$ بدلاً من المعادلة الأولى.

مثال: بالنسبة للحقل $GF(2^3)$ لدينا الجدول التالي:

		000	001	010	011	100	101	110	111
	$\times$	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
010	x	0	x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
011	$x+1$	0	$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
100	x^2	0	x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
101	x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
110	x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
111	x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

ويكون بالتالي ناتج جداء **110** مع **101** هو ما يلي:

110 → $x^2 + x$
011 → $x + 1$
$(x^2 + x)(x + 1) = x^3 + 2x^2 + x$
= x3 + x   2 mod 2 = 0

(x3 + x) mod (x3 + x + 1) =   x3 + x
- x^3 + x + 1
- 1
= 1 -1 mod 2 = 1

عكوس قيم حقول غالواس Galois Field Inverses

عكس العدد **b** من أعداد الحقل $GF(2^n)$ هو العدد b^{-1} بحيث يكون لدينا: $b^{1} \times b = 1$ ضمن الحقل $GF(2^n)$.

مثال: الحقل $GF(2^3)$

b	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
b^{-1}	**none**	**001**	**101**	**110**	**111**	**010**	**011**	**100**

ملاحظة هامة:

يعتمد المعيار **AES** على استخدام الحقل $GF(2^8)$ كما سنبين لاحقاً مع ملاحظة ما يلي:

- كثير الحدود الأولي لهذا الحقل هو: $P_8 = x^8 + x^4 + x^3 + x + 1$.
- تستخدم المرحلة **SubBytes** حسابات هذا الحقل كأساس لعمل صناديق الاستبدال **S-Boxes** الخاصة بها.
- تستخدم المرحلة **MixColumns** ضرب المصفوفات ضمن هذا الحقل.
- تعتمد عملية توليد مفاتيح الجولات على إضافة خانات عشوائية إلى مفتاح كل جولة.

4. التحويلات Transformations

يستخدم المعيار **AES** أربعة أنواع من التحويلات، وهي:

1. الاستبدال **Substitution**.
2. التدوير **Permutation**.
3. الخلط **Mixing**.
4. إضافة المفتاح **Key-adding**.

والتي سنتحدث عنها.

الاستبدال Substitution

يختلف الاستبدال الخاص بالمعيار **AES** عن مثيله المستخدم مع المعيار **DES**، حيث ينفذ هنا على مستوى البايت **Byte** ويجري استخدام جدول واحد للتحويل وهذا يعني أنه:

> لو كان لدينا بايتين متماثلين فإن نتيجة التحويل عليهما ستكون واحدة. يجري تعريف التحويل بواسطة تعريف لعملية بحث ضمن جدول أو رياضياً وفقاً للعمليات الممكنة على الحقل **Field**: $GF(2^8)$.

التحويل SubBytes:

يستخدم هذا التحويل للاستبدال على مستوى البايت ونعبر عن البايت على شكل خانتين من خانات نظام العد الست عشري **2 hexadecimal digits**. تحدد الخانة اليسارية رقم السطر أما الخانة اليمنى فتحدد رقم العمود ضمن جدول الاستبدال الذي يتضمن ضمنه البايت الجديد الناتج عن التحويل. يجري تمثل الحالة ضمن هذا التحويل على شكل مصفوفة **4x4** من البايتات. وتتم

عملية التحويل بيت فبايت ضمن المصفوفة ولكن يبقى ترتيبها هو نفسه. يوجد إذاً 16 تحويل محتمل من بايت إلى بايت. يبين الشكل التالي جدول التحويل لصندوق الاستبدال **S-box** الخاص بالتحويل **SubBytes**:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
x	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

أما جدول التحويل العكسي الخاص بالتحويل **InvSubBytes** فهو مبين في الشكل التالي:

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
x	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

مثال:

1101 0101 → row 13, column 5

والتي يجري غالباً تمثيلها وفقاً لنظام العد الست عشري:

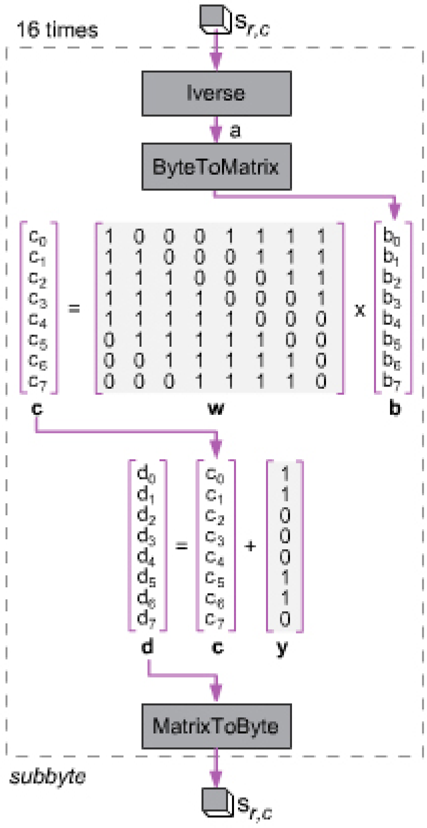
1011 0110 → row d, column 5

فتصبح النتيجة وفقاً لجدول التحويل:

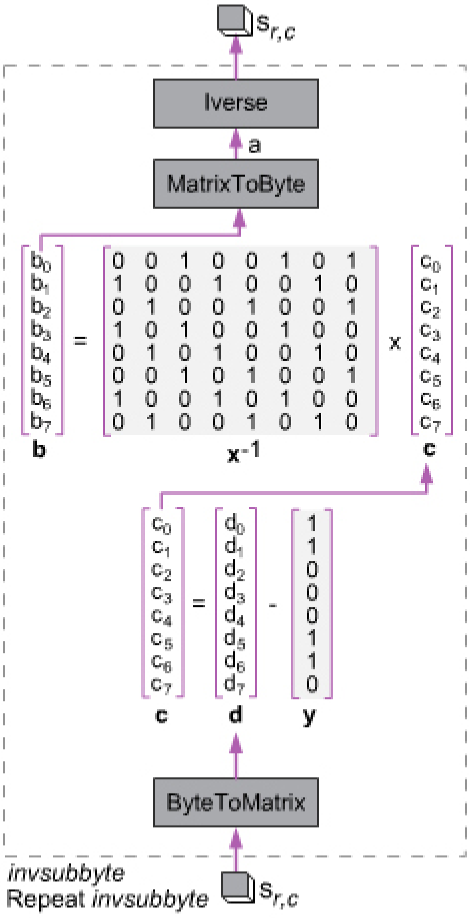
03 → 0000 0011

على الرغم من أنه يمكن استخدام الجدولين السابقين لإيجاد الاستبدال الممكن لكل بايت، فإن

AES يعرف عملية التحويل جبرياً باستخدام الحقل $GF(2^8)$ مع كثير الحدود الأولي لهذا الحقل $P_8 = x^8 + x^4 + x^3 + x + 1$ كما هو مبين في الشكل التالي:



يقوم التحويل **SubBytes** بتنفيذ التابع **subbyte** ست عشرة مرة. أما التحويل **InvSubBytes** فيقوم بتنفيذ التابع **invsubbyte** ست عشرة مرة. ويجري في كل مرة تحويل بايت واحد فقط.

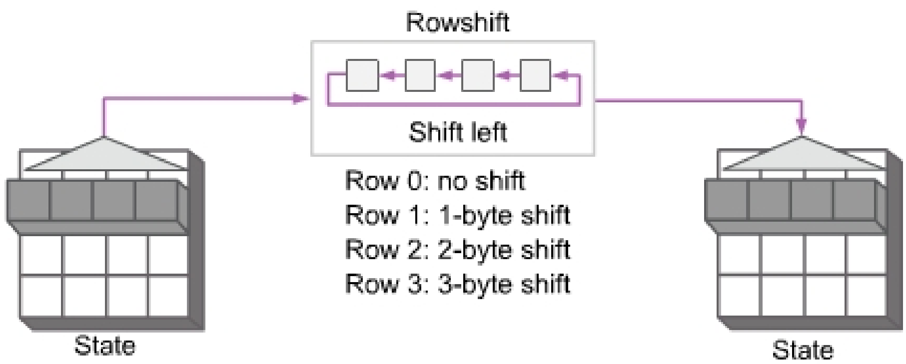


يمكن حساب العكس لكل بايت ضمن الحقل $GF(2^8)$ باستخدام كثير الحدود الأولي $P_8 = x^8 + x^4 + x^3 + x + 1$ كما شرحنا سابقاً. لاحظ أن عكس البايت 00_{16} هو نفسه. تصف بعدها البايتات المعكوسة ضمن مصفوفة عمودية **b** (من عمود واحد). يجري بعدها ضرب هذه المصفوفة مع مصفوفة مربعة ثابتة **X** لنحصل كنتيجة على المصفوفة العامودية **c**، التي نضيفها إلى المصفوفة العامودية الثابتة **y**. ينتج عن ذلك بايت جديد ضمن مصفوفة النتيجة **d**. لاحظ أن عمليات الضرب والجمع كلها تجري ضمن الحقل $GF(2)$ يقوم التابع **invsubbyte** بنفس العمليات بترتيب معاكس كما هو مبين في الشكل التالي:

التدوير Permutation

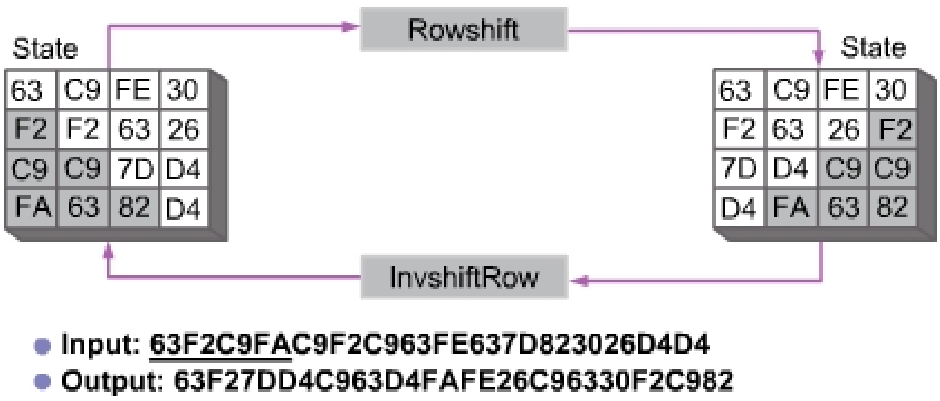
تجري عملية التدوير المرتبطة بالمعيار **AES** على مستوى البايت وليس على مستوى البت كما هو الحال في حالة المعيار **DES**. كما تتم هذه العملية بتنفيذ عمليات إزاحة **Shifting** لكامل البايت ولا يتغير ترتيب البت ضمنها.

- **التحويل ShiftRows**: يستخدم هذا التحويل للاستبدال أثناء عملية التشفير لتنفيذ إزاحات إلى اليسار. يرتبط عدد البايتات المزاحة (**0** أو **1** أو **2** أو **3**) برقم السطر ضمن مصفوفة الحالة. هذا يعني أنه لا تجري إزاحة السطر رقم **0**، في حين يزاح السطر الأخير ثلاثة بايتات يوضح الشكل التالي كيفية عمل تحويل الإزاحة:



يقوم تحويل الإزاحة العكسي **InvShiftRows** بتنفيذ إزاحات إلى اليمين بدلاً من اليسار ولنفس عدد البايات حسب رقم السطر في مصفوفة الحالة.

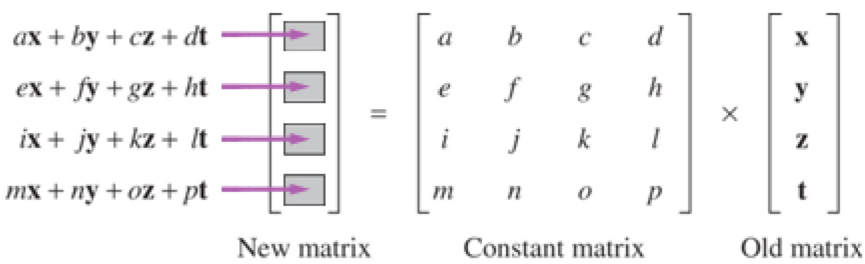
مثال:



الخلط Mixing

تعتمد عملية الاستبدال التي تحدثنا عنها فيما سبق والتي ينفذها التحويل **SubBytes** على تبديل قيمة البايت الأصلية استناداً على هذه القيمة وعلى قيمة موضع ضمن جدول، وليس لنتيجة العملية أي ارتباط بالبايتات المجاورة. يمكن القول أن هذه العملية تجري ضمن البايت نفسها **Intrabyte**.

أما عملية التدوير فهي تسعى إلى تبديل موضع البايتات دون أي تعديل ضمنها. نحتاج إذا إلى عملية لتعديل وتغيير محتوى البايتات من الدخل **Interbyte**. نحتاج إلى خلط البايتات على مستوى البت للتمويه أكثر. تجري عملية الخلط بأخذ 4 بايت بنفس الوقت لجمعها وإنشاء بايتات جديدة. لضمان أن تكون البايتات الأربعة الجديدة (حتى لو كانت البايتات الأصلية متماثلة) يجري ضرب البايتات الأصلية بقيمة ثابتة مختلفة، ثم تتم عملية خاطها. يوضح الشكل التالي الفكرة.



كما يبين الشكل التالي المصفوتين الثابتين المستخدمتين أثناء عمل التحويل **MixColumns** و**InvMixColumns**:

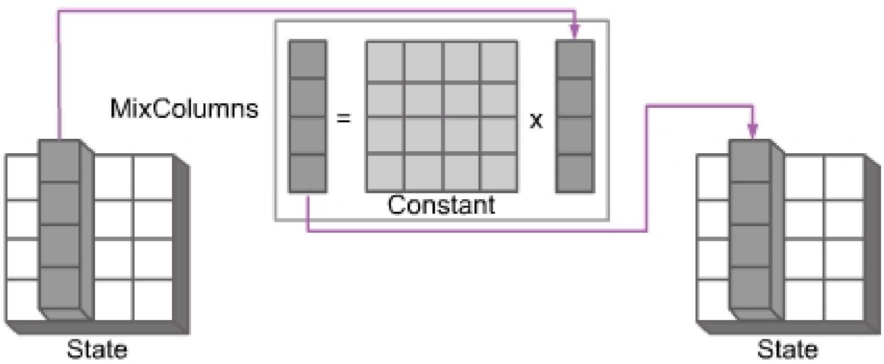
$$C = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \xleftrightarrow{\text{Inverse}} C^{-1} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

التحويل MixColumns:

تعمل هذ التحويل على مستوى الأعمدة ويجري تحويلاته على كل عمود من أعمدة مصفوفة الحالة. يتمثل التحويل بضرب مصفوفة الحالة بمصفوفة مربعة ثابتة. يجري التعامل مع البايتات المكونة لكلا المصفوفتين السابقتين كما لو أنها مؤلفة من **8** بت لكل منها أو كثيرات حدود مزودة بأمثال ضمن الحقل $GF(2)$. تجرى عملية الضرب للبايتات ضمن الحقل $GF(2^8)$ آخذين بعين الأعتبار أن كثير الحدود الأولي هو:

$$P_8 = x^8 + x^4 + x^3 + x + 1$$

أما الجمع فهو عملية **XOR** لكلمات مؤلفة من **8** بت. يوضح الشكل التالي طريقة تنفيذ التحويل **MixColumns**:



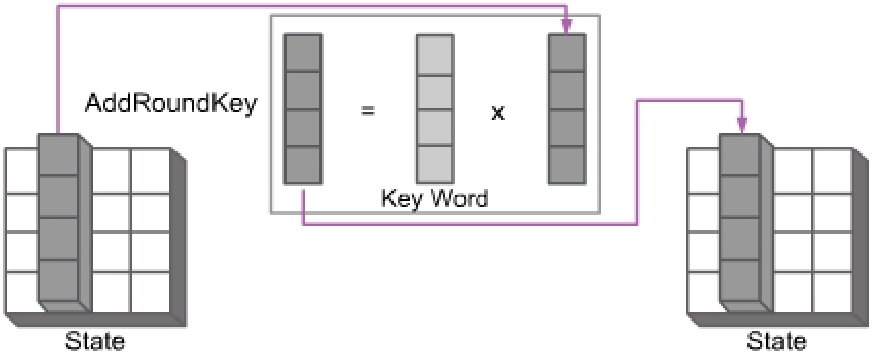
إضافة المفتاح Key Adding

وهو من أهم التحويلات المرتبطة بالمعيار **AES**. تستخدم كافة التحويلات الأخرى خوارزميات معروفة وقابلة للعكس. إذا لم يتم إضافة مفتاح التشفير إلى الحالة في كل جولة فإنه من السهل جداً فك التشفير والتعرف على النص المرسل انطلاقاً من النص المشفر. ويعتبر مفتاح التشفير هو السر الوحيد بين الطرفين الذين يتبادلا المعلومات.

يستخدم المعيار **AES** تقنية نشر المفتاح لإنشاء $N_r + 1$ مفتاح جولة انطلاقاً من مفتاح التشفير. يبلغ طول كل من مفاتيح الجولات **128** بت ويجري التعامل معه كما لو أنه مكون أربع كلمات كل واحدة منها مؤلفة من **32** بت .ومن أجل إضافة المفتاح إلى الحالة، يجري معاملة كل كلمة على أنها عمود مصفوفة.

التحويل AddRoundKey:

يشبه هذا التحويل التحويل **MixColumns** فهو يقوم بإضافة مفتاح الجولة إلى أعمدة مصفوفة الحالة بدلاً من عملية الضرب المنفذة مع التحويل **MixColumns**. وعلى اعتبار أن عملية الجمع والطرح هما عمليتان متمثلتان ضمن الحقل، فإن هذا عكس هذا التحويل هو نفسه. يبين الشكل التالي طريقة عمل التحويل **AddRoundKey**:



5. نشر مفتاح التشفير Key Expansion

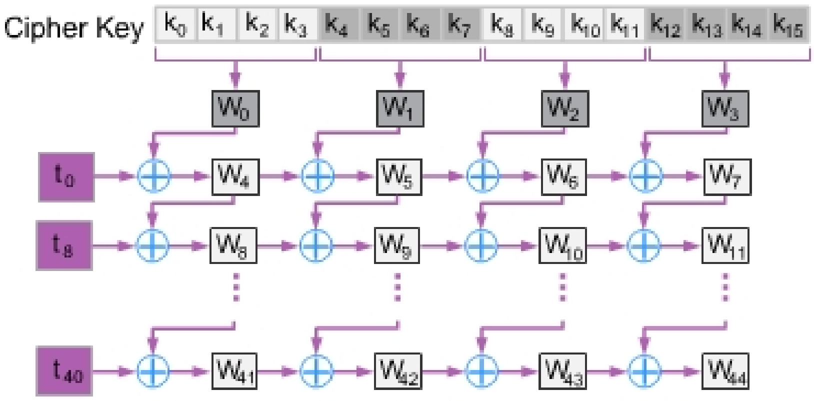
لإنشاء مفاتيح الجولات، يستخدم المعيار **AES** عملية النشر لمفتاح التشفير الرئيسي. إذا كان عدد الجولات مساوياً لـ N_r جولة، فإنه يجب إنشاء $N_r + 1$ مفتاح جولة بطول **128** بت كل منها انطلاقاً من مفتاح التشفير الأساسي بطول **128** بت أيضاً. يجري استخدام مفتاح الجولة الأول مع تحويل ما قبل الجولة **Pre-round transformation**. أما بقية المفاتيح فتستخدم مع التحويل الأخير **(AddRoundKey)** في نهاية كل جولة.

يقوم تابع نشر المفتاح بإنشاء مفاتيح الجولات كلمة فكلمة، حيث كل كلمة هي **4** بايت. أي أن هذا التابع يقوم بإنشاء $4 \times (N_r + 1)$ كلمة ونرمز لها كما يلي:

$$w_0, w_1, \ldots, w_{4(N_r+1)-1}$$

أي لدينا بالنسبة للنسخة **AES-128** (**10** جولات) **44** كلمة وبالنسبة للنسخة **AES-192** (**12** جولة) **52** كلمة وبالنسبة للنسخة **AES-256** (**14** جولة) **60** كلمة.

يبين الشكل التالي كيفية بناء **44** كلمة من أجل النسخة **AES-128** من المفتاح الأصلي:



تجري هذه العملية وفقاً لما يلي:

1. نأخذ الكلمات الأربعة (w_0, w_1, w_2, w_3) من مفتاح التشفير.
2. يتم حساب باقي الكلمات (w_i من أجل قيم **i=1** حتى **43**) وفقاً لما يلي:
   - إذا كان لدينا $i \bmod 4 \neq 0$ فإن $w_i = w_{i-1} \oplus w_{i-4}$.
   - إذا كان لدينا $i \bmod 4 = 0$ فإن $w_i = t \oplus w_{i-4}$ حيث يتم حساب **t** وفقاً للمعادلة التالية:

$$t = SubWord(RotWord(w_{i-1})) \oplus RCon_{i/4}$$

مع العلم أن:

- **RotWord** هو تابع دوران مشابه للتحويل **ShiftRows** ولكن يجري تطبيقه هنا على سطر واحد. يأخذ التابع كلمة على شكل شعاع (مصفوفة وحيدة البعد) من أربعة بايتات ويزيح كل بايت إلى اليسار دورانياً.
- **SubWord** هو تابع استبدال مشابه للتحويل **SubBytes** ولكن يجري تطبيقه هنا على على أربعة بايت فقط. يأخذ التابع بايت من الكلمة ويبدلها ببايت آخر.
- **RCon** هي مجموعة قيم ثابتة. يوجد لكل جولة قيمة ثابتة من **4** بايت بحيث تكون البايتات الموجودة في أقصى اليمين مساوية دوماً للصفر. يوضح الجدول التالي القيم الثابتة من أجل النسخة **AES-128**.

round	RCon
1	01 00 00 00
2	02 00 00 00
3	04 00 00 00
4	08 00 00 00
5	10 00 00 00
6	20 00 00 00
7	40 00 00 00
8	80 00 00 00
9	1B 00 00 00
10	36 00 00 00

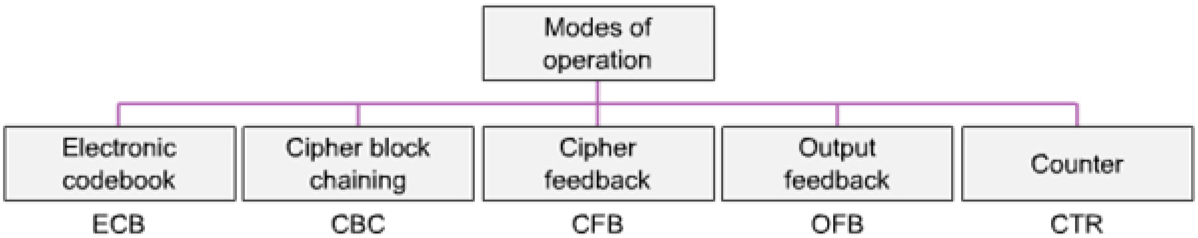
6. التشفير باستخدام مشفرات المفتاح المتناظر الحديثة

Encipherment Using Symmetric-Key Ciphers

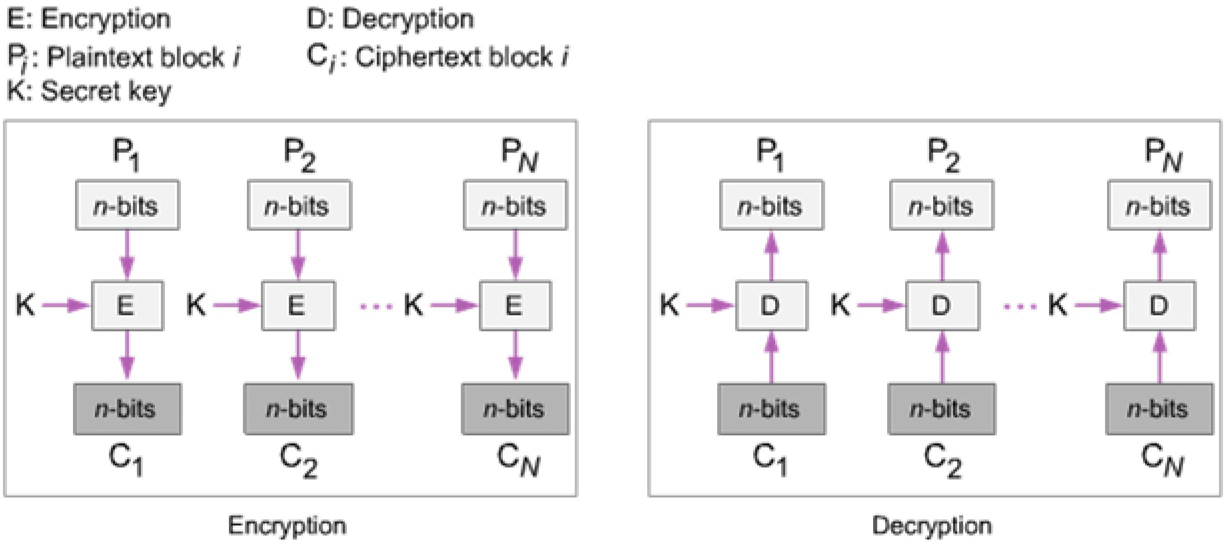
يمكن تنفيذ عملية تشفير بمفتاح متناظر باستخدام مشفرات كتلية حديثة. تحدثنا فيما سبق عن معيارين حديثين للمشفرات الكتلية الحديثة، هما:

DES و**AES** اللذين صمما للتعامل مع كتل من حجم ثابت.

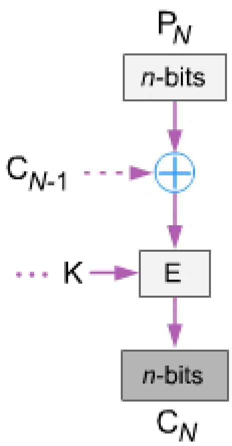
لكن عند التعامل مع النصوص الحقيقية، فهي من حجوم متغيرة وهي حتماً أكبر من **64** أو **128** بت. يبين الإيضاح التالي خمسة طرق للعمل **Modes of operation** واستخدام أحد المعيارين السابقين.



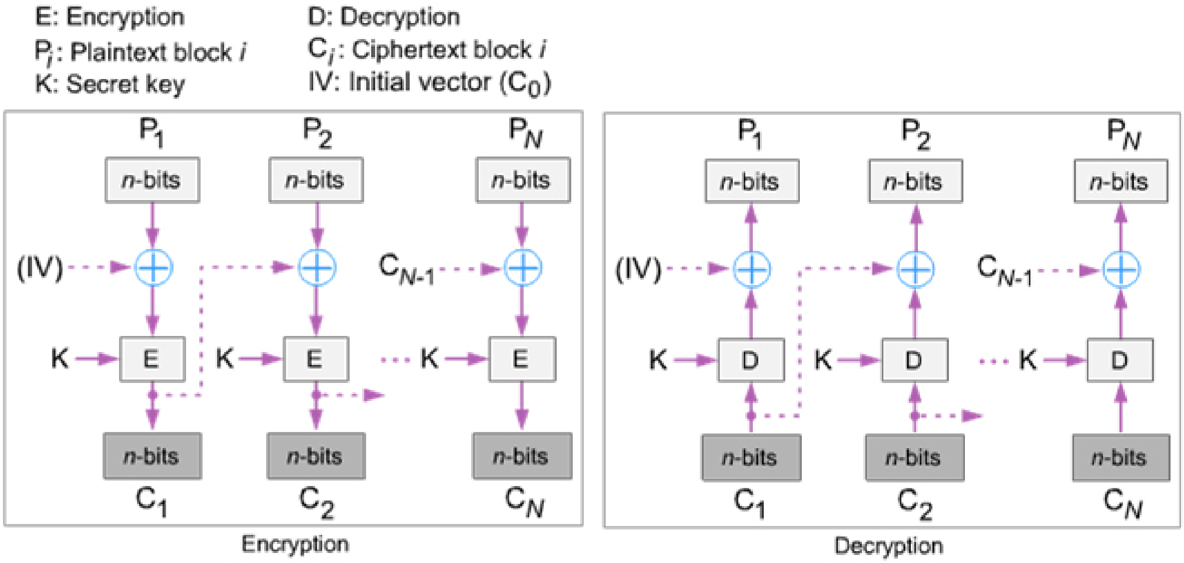
- تعتبر طريقة العمل Electronic codebook (ECB) أبسط طريقة عمل حيث يجري تقسيم النص غير المشفر إلى N كتل من الحجم n بت ونستخدم نفس مفتاح التشفير وفك التشفير مع كافة الكتل، كما هو مبين في الشكل التالي:



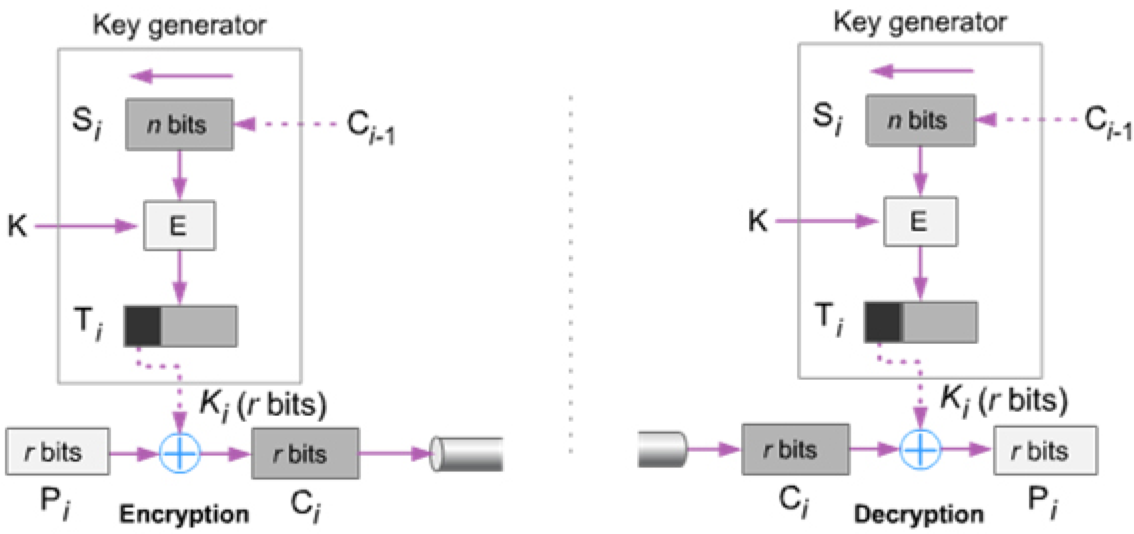
- في طريقة العمل Cipher block chaining (CBC)، يتم تطبيق التابع XOR على كتلة من النص غير المشفر مع الكتلة السابقة من النص المشفر قبل البدء بتشفير الكتلة فعلياً.



بعد تشفير الكتلة، فإنها ترسل ولكن يجري الاحتفاظ بنسخة عنها لاستخدامها مع تشفير الكتلة التالية. يتفق طرفا الاتصال حول الإعدادات الابتدائية لتطبيق التابع XOR عليها وعلى أول كتلة من النص المشفر. تجري عملية التشفير وفك التشفير هما كما مبين في الشكل التالي:

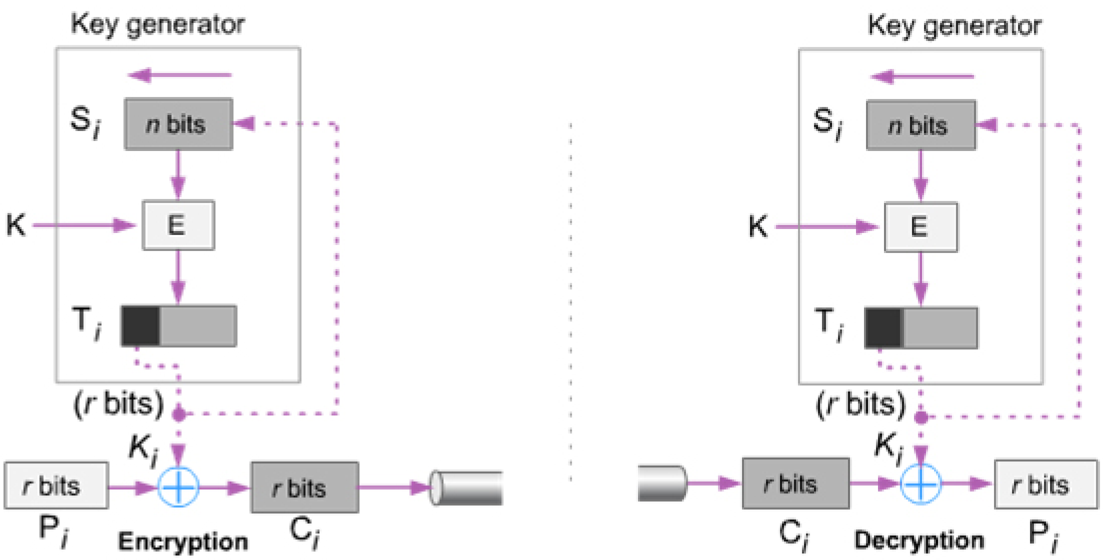


- لتشفير كمية قليلة من المعطيات بالزمن الحقيقي، يمكن استخدام الطريقة **Cipher feedback (CFB)**. تستخدم طريقة العمل هذه مشفر كتلي معياري (مثل **DES** أو **AES**) لتشفير سجل إزاحة ولكن يجري استخدام العملية **XOR** لتشفير وفك تشفير وحدات المعطيات الحقيقية. تستخدم الطريقة **CFB** المشفرات الكتلية ولكن النتيجة هي مشفر دفق معطيات حيث يجري تشفير كل وحدة معطيات بمفتاح مختلف، كما هو مبين في الشكل التالي:

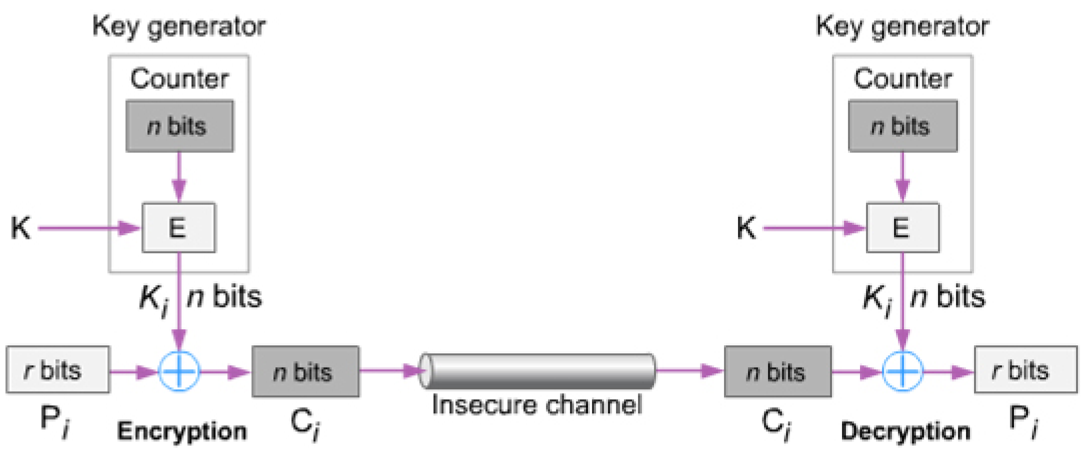


- تشبه طريقة العمل **(OFB) Output feedback** كثيراً طريقة العمل **CFB** مع اختلاف وحيد. لا ترتبط كل خانة (بت) من خانات النص المشفر بالخانة أو بالخانات السابقة. هذا يمنع انتشار الأخطاء. بدلاً من استخدام كتلة النص المشفر السابقة، يقوم **OFB** باستخدام

المفتاح السابق كتغذية راجعة للمشفر. يوضح الشكل التالي طريقة عمل **OFB**:



- في طريقة العمل **Counter (CTR)**، لا يوجد تغذية خلفية. يمكن تحقيق العشوائية في مفتاح دفق المعطيات باستخدام عداد من **n** بت. يأخذ هذا العداد قيمة ابتدائية معرفة مسبقاً **(IV)** وتجري زيادة قيمته وفقاً لقاعدة مسبقة التعريف. يوضح الشكل التالي طريقة عمل **CTR**:



لتشفير وحدات صغيرة من المعطيات، مثل الحروف والبتات، جرى إعادة تصميم عدة مشفرات دفق معطيات من الأساس. تكون هذه المشفرات فعالة لمعالجات الزمن الحقيقي. من بين هذه المشفرات لدينا: **RC4** و **A5/1**.

- **RC4** هو مشفردفق معطيات على مستوى البايت. يقوم هذا المشفر بتطبيق التابع **XOR**

على كل بايت من النص غير المشفر مع بايت من المفتاح لحساب بايت من النص المشفر. يمكن للمفتاح السري الذي نستخدمه لانتاج مفاتيح بطول بايت واحدة عند تشفير دفق المعطيات، أن يتضمن من **1** إلى **256** بايت. يعتمد مولد دفق المفاتيح على تدوير حالة مؤلفة من **256** بايت.

- **A5/1** هي مشفر دفق معطيات مستخدم الاتصالات التلفونية المحمولة. يقوم **A5/1** بإنشاء دفق من الخانات من مفتاح بطول **64** بت.

الجزء الثاني:

التشفير بمفتاح غير متناظر

Asymertic-Key Encipherment

الفصل الخامس

مقدمة في التشفير بمفتاح غير متناظر

Introduction to Asymmetric-Key Cryptography

الأهداف Objectives

يهدف هذا الفصل إلى التعريف بما يلي:

- عرض نبذة تاريخية عن أساليب التشفير بمفتاح غير متناظر.
- تعريف بالبنية الأساسية لأساليب التشفير بمفتاح غير متناظر المعتمدة على التوابع وحيدة الاتجاه المزودة بمصيدة **Trapdoor One-Way Functions**.
- وصف طريقة عمل نظام التشفير "حقيبة الظهر" **Knapsack Cryptosystem**.
- وصف طريقة عمل نظام التشفير **RSA**.

1. مقدمة Introduction

تتواجد أساليب التشفير بمفتاح متناظر وأساليب التشفير بمفتاح غير متناظر جنباً إلى جنب. ونعتقد أن كل منها يكمل الآخر وتغطي مزايا كل منها سيئات الآخر. يكمن الفرق الجوهري بينهما في كيفية محافظتهما على سرية المعلومات. تتميز أنظمة التشفير بمفتاح غير متناظر بأن كل طرف يحتفظ لنفسه بمفتاحه السري ولا يشارك به أي طرف آخر.

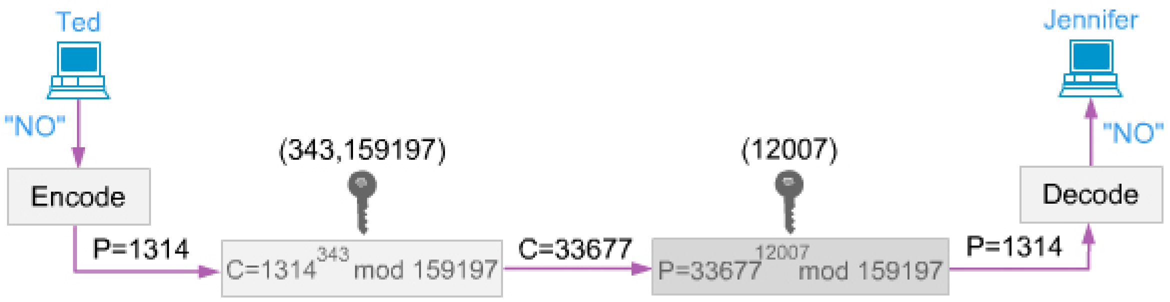
في مجتمع مؤلف من **n** شخص، نحتاج إلى وجود $\frac{n(n-1)}{2}$ مفتاح مشترك من أجل التعامل باستخدام أساليب تشفير بمفتاح متناظر. أما في حالة التعامل باستخدام أساليب تشفير بمفتاح غير متناظر، فإننا نحتاج إلى **n** سر شخصي فقط.

نحتاج إلى أساليب التشفير بمفتاح غير متناظر من أجل بعض الأدوات الأمنية والتي نذكر منها:

- التحقق من الهوية **Authentication**.
- التوقيع الالكتروني **Digital Signature**.

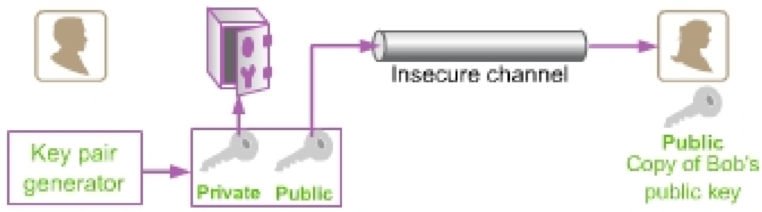
ولكن يمكن أن نقول بشكل عام، أنه عند الحاجة إلى مفتاح سري شخصي فيمكن استخدام أساليب التشفير بمفتاح غير متناظر.

تعتمد أساليب التشفير بمفتاح غير متناظر على تطبيق توابع رياضية عددية لأنه مع هذه الأساليب ننظر إلى النصوص غير المشفرة والمشفرة على أنها مجموعة من الأعداد ونستخدم التوابع الرياضية على هذه الأعداد لتوليد أعداد أخرى، كما هو مبين في الشكل التالي:

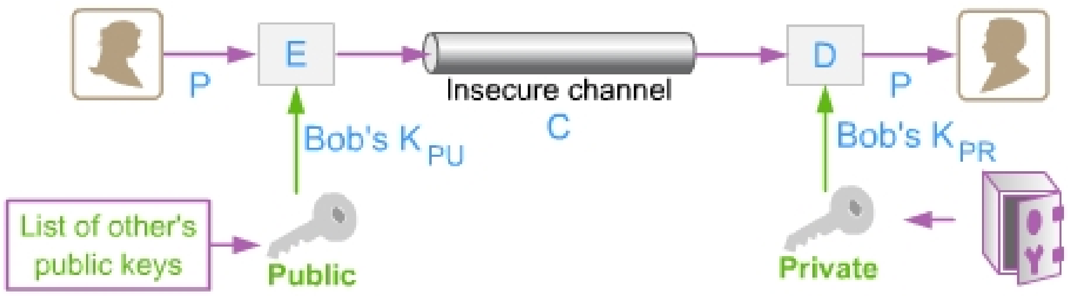


تعتمد أساليب التشفير بمفتاح غير متناظر على وجود مفتاحين: مفتاح عام **Public key** (K_{PU}) ومفتاح خاص **Private key** (K_{PR}). وتعمل هذه الأساليب وفقاً للمبدأ التالي:

> أي نص يجري تشفيره بمفتاح عام لشخص لا يمكن فك تشفيره إلا بالمفتاح الخاص الذي يعرفه الشخص ذاته فقط. يوضح الشكل التالي هذه الفكرة:



كما يوضح الشكل التالي طريقة عمل أساليب التشفير بمفتاح غير متناظر:



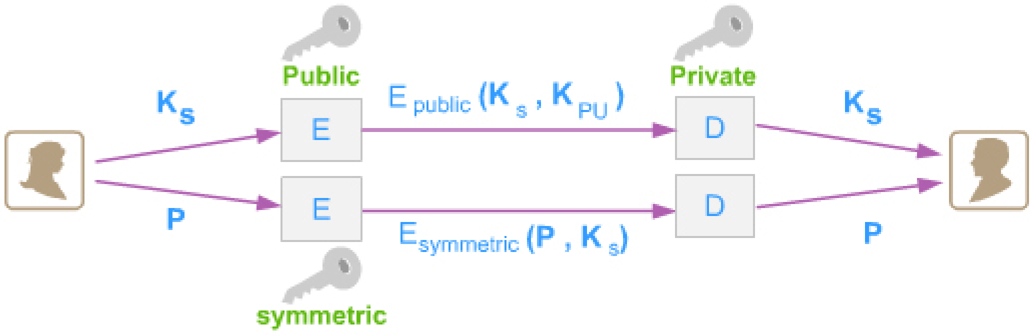
ولدينا:

$$E = E(K_{PU}, P)$$
$$P = D(K_{PR}, E)$$

2. المفاتيح العامة والمتناظرة Public and Symmetric Keys

في واقع الأمر نحتاج إلى كلا النوعين من المفاتيح العامة والمتناظرة ولكن تتميز المفاتيح العامة ببطئها الشديد في تشفير كامل الرسالة المطلوب تشفيرها. لحل هذه المشكلة يمكن اللجوء إلى المفاتيح المتناظرة ولكن نحتاج إلى تبادلها بشكل آمن بين الطرفين المتصلين.

يمكن تبادل المفتاح المتناظر باستخدام تقنية المفاتيح العامة. ثم نستخدم خوارزمية مفتاح متناظر سريعة (مثل **RC4**) لتبادل الرسائل بصورة آمنة. يبين الإيضاح التالي هذه الفكرة:



3. التوابع وحيدة الاتجاه ذات المصيدة Trapdoor One-Way Functions

نقول عن تابع $y = f(x)$ (ذو التابع العكسي $x = f^{-1}(y)$) أنه وحيد الاتجاه **One- way**، إذا تحقق لدينا الشروط التالية:

1. إذا توفر لدينا x فإنه من السهل حساب $y = f(x)$.
2. أما إذا توفر لدينا y فإنه لا يمكن حسابياً تحديد قيمة $x = f^{-1}(y)$.

مثال: تابع حساب العوامل **Factroing**:

إذا كان العددين **p** و**q** هما عددان أوليان كبيران فإنه من السهل حساب القيمة $n = p \times q$ ولكن من غيرالممكن حساب عوامل **n** المتمثلة في العددين **p** و**q** (يجب تجربة كافة القيم المرتبطة بهما!).

نقول عن تابع وحيد الاتجاه $y = f(x)$ أنه مزود بمصيدة **Trapdoor**، إذا تحقق لدينا ما يلي:

1. التابع $y = f(x)$ هو تابع وحيد الاتجاه.
2. يوجد مصيدة سرية تسمح لنا بحساب $x = f^{-1}(y)$ بسهولة.

مثال: تابع حساب العوامل **Factroing**:

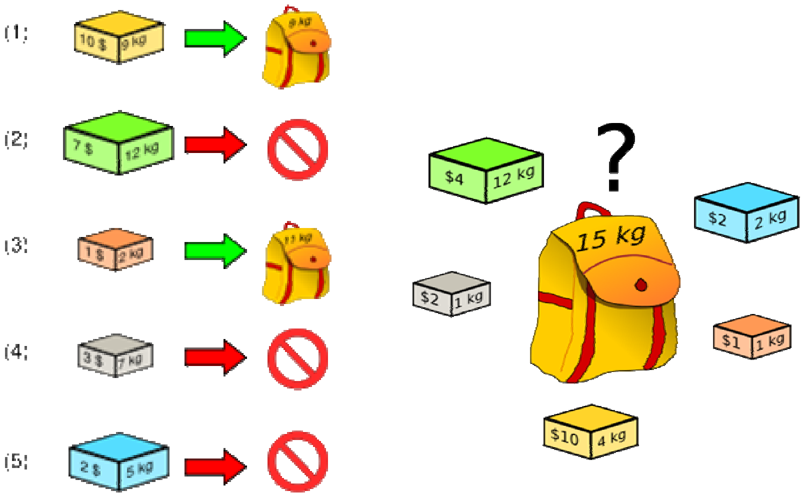
إذا كان العددين **p** و**q** هما عددان أوليان كبيران فإنه من السهل حساب القيمة $n = p \times q$ ولكن من غيرالممكن حساب عوامل **n** المتمثلة في العددين **p** و**q**. أما إذا علمنا قيمة **q** فإنه من السهل تحديد قيمة **p** (**q** هي المصيدة لحساب **p** من **n**).

نحتاج إلى مفهوم التوابع وحيدة الاتجاه ذات المصيدة بسبب كونها تشكل جوهر عمليات التشفير بالمفاتيح العامة. يجب أن يكون تابع التشفير $C = E(K_{PU}, P)$ تابعاً وحيد الاتجاه (يجب أن يكون غير ممكن حساب P أو K_{PU} انطلاقاً من C). كما يجب أن يتوفر لدينا مصيدة من

أجل فك تشفير النص المرسل وحتى نتمكن بسهولة من حساب P انطلاقاً من C بعد معرفة المصيدة التي تمثل هنا المفتاح الخاص K_{PR}.

4. نظام تشفير "حقيبة الظهر" Knapsack Crypytosystem

يعتبر هذا النظام أول نظام تشفير بمفتاح عام وقد من قبل Merkle وHellamn عام 1978. يعتمد هذا النظام على فكرة إيجاد حل لمسألة تعبئة حقيبة الظهر التي تقول ما يلي: إذا كان لدينا حقيبة بسعة مساوية لـ s وكان لدينا k غرض، فالسؤال المطروح هو: ما هي مجموعة الأغراض الواجب اختيارها لملئ الحقيبة تماماً؟



الوصف الرياضي للمسألة Mathematical Description

إذا كان حجم الغرض i هو a_i

يكون الغرض i ضمن الحقيبة، إذا كانت القيمة $x_i = 1$

والغرض i غير موجود ضمن الحقيبة، إذا كانت القيمة $x_i = 0$

الحجم الكلي للأغراض ضمن الحقيبة هو: $x_1 a_1 + x_2 a_2 + \ldots + x_k a_k$.

مثال:

- **a = [9, 12, 2, 7, 5]**
- **s = 11**

- x = [1, 0, 1, 0, 0]
- Sum = 1×9 + 0×12 + 1×2 + 0×7 + 0×5 = 11

- **التابع وحيد الاتجاه:** إذا كان لدينا *x* فإنه من السهل حساب مجموع حجوم الأغراض ضمن الحقيبة. أما إذا كان لدينا حجم الحقيبة s فإنه من الصعب تحديد صيغة ملائمة لـ x لتحديد الأغراض التي تملئ تماماً الحقيبة: يوجد 2^k صيغة ممكنة ويجب تجربتها كلها لتحديد الحل المناسب.
- **التابع وحيد الاتجاه ذو المصيدة:** إذا كانت حجوم الأغراض ممثلة بشكل متزايد وكان لدينا $a_i \geq a_1 + a_2 + \ldots + a_{i-1}$ Superincreasing tuple فإنه يمكن استخدام الخوارزمية التالية لحل المسألة:

```
for (i = k down to 1) {
        if(s>=ai){
      xi = 1
      s = s – ai
   }
   else xi = 0
 }
```

مثال:

- a = [2, 3, 6, 12, 25, 50, 100, 200]
- s = 139
- Steps:
  - 139 < 200 → x8 = 0
  - 139 > 100 → x7 = 1 s = 39
  - 39 < 50 → x6 = 0
  - 39 > 25 → x5 = 1 s = 14
  - 14 > 12 → x4 = 1 s = 2
  - 2 < 6 → x3 = 0
  - 2 < 3 → x2 = 0
  - 2 = 2 → x1 = 1 s = 0

وتشمل المراحل التالية:

1. توليد المفتاح.
2. التشفير.
3. فك التشفير.

توليد المفتاح Key Generation

- نقوم بإنشاء مجموعة متزايدة $b=[b_1,b_2,\ldots,b_k]$ محققة للشرط التالي: $b_i \geq b_1+b_2+\ldots+b_{i-1}$ ستكون هذه المجموعة جزءاً من المفتاح الخاص على اعتبار أنها تساعد على حل المسألة. لكن لدينا مشكلة تخبئة هذه المجموعة؟ يمكن تحقيق ذلك من خلال تدويرها واستخدام التابع **mod** ليصبح من المستحيل ترتيبها بشكل متزايد.
- نختار بعدها قيمة لـ ***n*** بحيث تحقق الشرط التالي: $n>b_1+b_2+\cdots+b_k$. أي أنها أكبر من حجم أي حقيبة للظهر. مثال: **n = 25**.
- نختار بعدها القيمة $r<n$ بحيث تكون أولية نسبياً معها **Relatively Prime** أي:
  - ليس لهما قاسم مشترك.
  - يجب أن تختلف نتيجة العملية $(b\times r)\bmod n$ عنهما.
  - مثال: **r = 7**.
- نحسب بعدها المصفوفة ***t*** بحيث يكون لدينا: $t_i=b_i\times r \bmod n$.
  - مثال: ***b* = [2, 3, 6, 12] → *t* = [14, 21, 17, 9]**
- ننشئ تدوير **Permutation** لـ **t** ونستخدمه لبناء ***a***. مثال: التدوير **[3, 2, 4, 1]**
  - **t = [14, 21, 17, 9] → a = [17, 21, 9, 14]**
- لا يمكن لمن يرغب بمعرفة التشفير أن ينجح في معرفة القيم الأصلية لـ **b** بدون معرفة **n** و**r**.
- ينتج عما سبق أن: المفتاح العام هو **a** والمفتاح الخاص هو **b** و**n** و**r** والتدوير.

التشفير Encryption

إذا أردنا تشفير القيمة **1001** فإن ذلك يتم كما يلي:

- **x = [1, 0, 0, 1]**.
- **a = [17, 21, 9, 14]** وهذا يؤدي إلى أن حجم الحقيبة هو: **17×1 + 21×0 + 9×0 + 14×1 = 31**.
- النص المشفر = **31**.

فك التشفير Decryption

يستقبل الطرف الثاني القيمة **s = 31** ونتعامل معها على الشكل التالي:

- نحسب القيمة $r^{-1} \bmod n$ لنستخدمها لحساب عكس نتيجة الضرب بـ **r**:

r=7,n=25
7^{-1} mod 25 = 18 (7×18 = 126, 126 mod 25=1)

- نحسب القيمة $s' = r^{-1} \times s \bmod n$:

s` = 18 × 31 = 558 mod 25 = 8

- نعكس عملية ملئ حقيبة الظهر **inv_knapsackSum** فنحصل على ما يلي:

b = [2, 3, 6, 12]

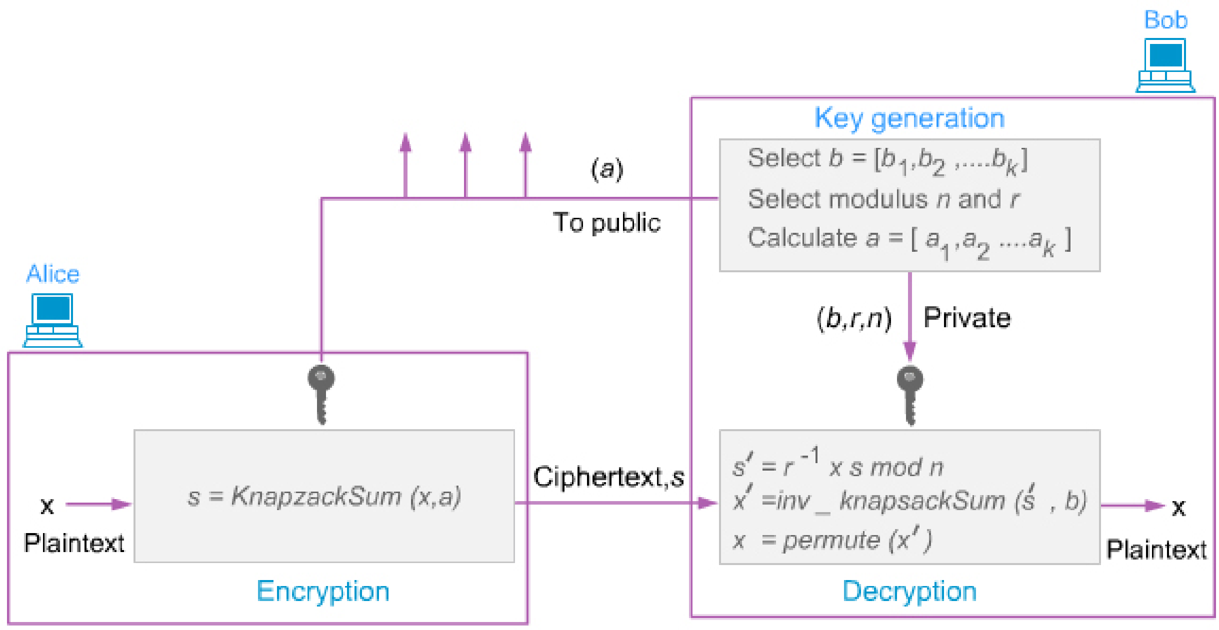
8 = 1 × 2 + 0 × 3 + 1 × 6 + 0 × 12 → 1010

- نقوم بتنفيذ عملية التدوير لنحصل على النص غير المشفر:

Permutation: [3, 2, 4, 1]

1 0 1 0 → 1 0 0 1

يلخص الشكل التالي مجمل العمليات التي تحدثنا عنها فيما سبق:



5. نظام التشفير RSA Crypytosystem RSA

يعد هذا النظام من أوائل وأكثر نظم التشفير المعتمدة على المفاتيح غير متناظرة استخداماً. تم تطويره للاستخدام العام في عام **1977** من قبل:

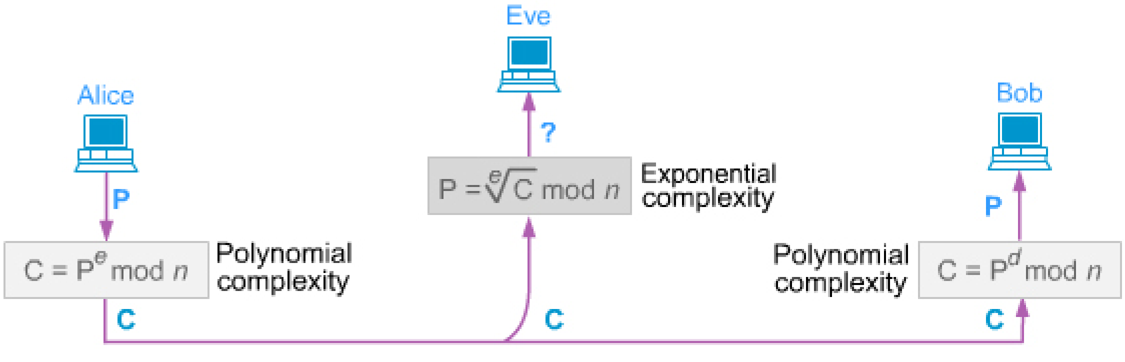
Ron Rivest, Adi Shamir, and Leonard Adleman في المعهد **MIT**.

يجري حالياً استخدام هذا النظام بكثرة وهو يشكل جزءاً لا يتجزأ من أنظمة التشفير المستخدمة في يومنا هذا، مثل: SSH وPGP وغيرها.

خوارزمية RSA RSA Algorithm

تعتمد خوارزمية هذا النظام على استخدام تابع حساب باقي قوة عدد **Modular Exponentiation Function**. ويتعامل مع قيم **mod n** حيث اختيار **n** من قيم كبيرة (قيم ممثلة على **1024** بت على الأقل أي ما يعادل تقريباً أرقام مكتوبة على **309** خانة عشرية).

- **التشفير Encryption**: تتم عملية التشفير وفقاً للمعادلة التالية : $C = P^e \mod n$ وهي معادلة سهلة الحساب (يمكن حسابها بزمن كثير حدود) حيث لدينا:
  - النصان **P** و**C** هما أرقام صحيحة ضخمة.
  - المفتاح العام هو **n modulus** والقوة **e**.
- **التابع وحيد الاتجاه One-Way Function**: لا يمكن للتابع $P = \sqrt[e]{C} \mod n$ أن يساعد على كسر الشيفرة ومعرفة النص الأصلي **P** انطلاقاً من **C**. لتنفيذ ذلك يجب فحص كافة الحالات $P < n$.
- **المصيدة من أجل فك التشفير Trapdoor for decryption**: يوجد القيمة **d** بحيث يكون لدينا $P = C^d \mod n$ ولدينا أيضاً:
  - القيمة **n** هي ناتج جداء العددين الأوليين **p** و **q** كما سنبين لاحقاً.
  - القيمة **d** هي المفتاح الخاص. يمثل الشكل التالي طريقة عمل خوارزمية **RSA**:



توليد مفتاح RSA RSA Key Generation

تتم عملية توليد مفتاح الخوارزمية **RSA** وفقاً للخطوات التالية:

- نقوم باختيار عددين أوليين كبيرين **p** و **q** يمكن تمثيلهما على **512** بت على الأقل

(عدد مكون من **154** خانة عشرية على الأقل).

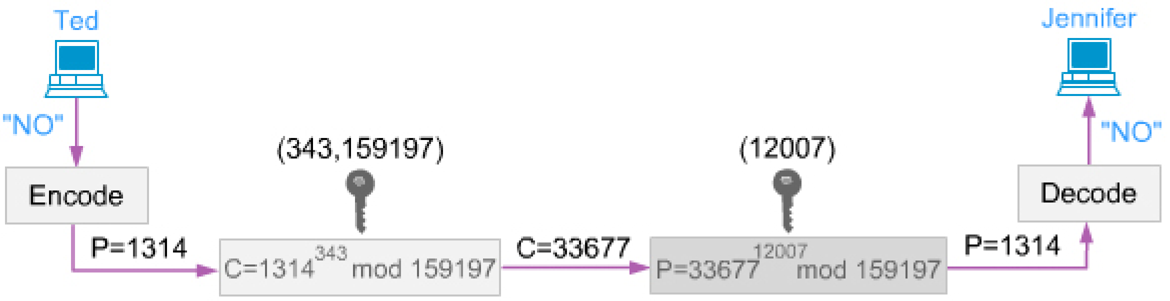
- نحسب القيمة $n = p \times q$.
- نحسب بعدها القيمة $\Phi(n) = (p-1) \times (q-1)$ حيث لا يمكن أن نحسب هذه القيمة بدون حساب لعاملين الأوليين للعدد **n** وهما: **p** و **q**. ويعتبر هذا الأمر أساسياً لأن عملية ضرب الأعداد الأولية الكبيرة هي عملية وحيدة الاتجاه.
- نقوم بعدها باختيار عدد **e** بحيث يكون لدينا:
  - $1 < e < \Phi(n)$.
  - **e** عدد أولي بالنسبة للعدد $\Phi(n)$.
- نحسب القيمة $d = e^{-1} \bmod \Phi(n)$ لنحصل على $e \times d \mod \Phi(n) = 1$.
- نحصل بالنتيجة على: المفتاح العام **e** و**n**. أما المفتاح الخاص فهو: **d**.

مثال لطريقة عمل الخوارزمية RSA RSA Example

Public key: n = 159197 (from 397 × 401)

e = 343 ← في الواقع تعتبر هذه القيمة صغيرة

Private Key: D = 12007 = 343 - 1 mod 158400



الجزء الثالث

التكامل، تحديد الهوية، وإدارة المفاتيح

Intergrity, Authentication and Key Management

الفصل السابع

تكامل الرسائل وتحديد الهوية

Message Integrity and Message Authentication

الأهداف Objectives

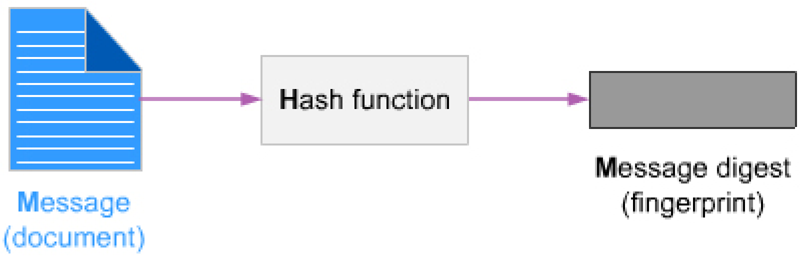
يهدف هذا الفصل إلى التعريف بما يلي:

- تعريف بأسلوب تكامل الرسائل.
- تعريف كيفية تحديد هوية المرسل.

1. تكامل الرسائل Message Integrity

يوجد حالات يكون فيها من المفيد أن نهتم بتكامل المعطيات أكثر من سريتها. يمكن مثلاً لشخص أن يكتب وصيته لتوزيع ممتلكاته بعد موته. لا حاجة لتشفير الوصية ولكن بعد موت الشخص نحتاج إلى تفحص الوصية والتحقق منها. يجب الحفاظ على تكامل الوصية لأن المطلوب حفظها دون أن يمسها أي تغيير.

يمكن الحفاظ على تكامل المعطيات من خلال البصمة المرتبطة بها **Fingerprint**. يجري رقمياً بناء البصمة الالكترونية **Message Digest** للحفاظ على محتوياتها بواسطة توابع تهشير **Hash Functions**. تقوم هذه التوابع بإنشاء صورة مضغوطة عن الرسالة مشابهة لبصمة الأصبع. يوضح الشكل التالي كيفية تنفيذ هذه العملية:



يتميز هذا الاسلوب في أن لكل رسالة بصمة الكترونية خاصة بها. تتمتع هذه البصة بأن حجمها أصغر بكثير من حجم الرسالة، ولكن يجب الحفاظ عليها ضد عمليات التغيير وتخزينها في مكان آمن.

التحقق من التكاملية Checking Integrity

للتحقق من تكاملية رسالة، نحتاج إلى تشفير تابع التهشير عليها مرة أخرى لإعادة توليد بصمتها الالكترونية ومقارنتها مع البصمة القديمة. إذا تطابقت البصمتان فهذا يعني أن الرسالة لم تتعرض لأية عملية تغيير.

كما يمكن تطبيق نفس المعالجة بين طرفين يتبادلان الرسائل ويحتاج كل منهما التأكد من الرسائل التي تصله فيما إذا وصلت كما أرسلت دون أي تعديل. لكن يجب أولاً ضمان وصول بصمة الرسالة من الطرف الأول إلى الثاني بشكل آمن قبل أن يقوم بدوره بإعادة حساب بصمة الرسالة مرة أخرى ومقارنته مع البصمة المرسلة من الطرف الأول مسبقاً.

معايير تابع التهشير Cryptography Hash Function Criteria

إذا كان لدينا تابع التهشير **h** وكانت لدينا الرسالة **M** فإن بصمة هذه الرسالة هي: **y=h(m)**.
لضمان تكامل الرسالة، يجب أن يتمتع تابع التهشير المستخدم بثلاث خصائص، هي:

- لا يمكن إيجاد رسالة أخرى **'M** بحيث يكون لدينا: **y=h(M')**. نسمي هذه الخاصة **Preimage Resistance**.
- لا يمكن تزوير الرسالة بسهولة. فعندما يقوم شخص بإرسال رسالة مع بصمتها الالكترونية إلى شخص ثاني، فلا يمكن للشخص الثاني أن ينشئ رسالة أخرى لها نفس البصمة الالكترونية. نسمي هذه الخاصة **Second Preimage Resistance**.
- لا يمكن أبداً إنشاء رسالتين مختلفتين لهما نفس البصمة الالكترونية. نسمي هذه الخاصة **Collision Resistance**.

حل مشاكل تكامل الرسائل Solving Integrity Problems

يساعد تقنية تكامل الرسائل على حل المشاكل التالية:

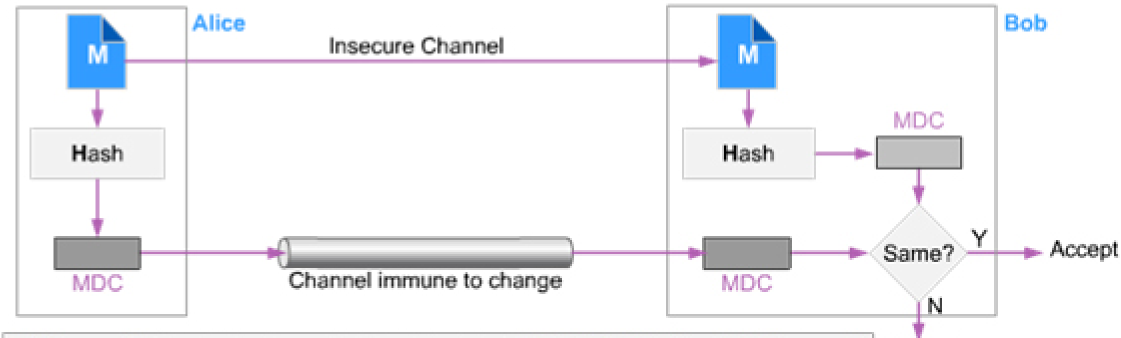
- **تعديل المحتوى Content Modification**: عند تعديل محتوى رسالة سيؤدي ذلك إلى توليد بصمة الكترونية مختلفة عن بصمتها السابقة وبالتالي سيتم اكتشاف عملية التعديل بسهولة.
- **تحديد الهوية Authentication**: لا يمكن لطرف ثالث أن يرسل رسالة مزورة لأنه لا يمكن له توليدها وفقاً لبصمة الرسالة الأصلية.
- **تعديل الوقت Timing Modification**: يمكن للمرسل أن يضيف إلى رسالته ختم زمني **Timestamp** بحيث لا يمكن لأي طرف ثالث تعديل الرسالة وتوليد بصمة الكترونية لها مماثلة للرسالة الأصلية.

2. التحقق من الهوية Message Authentication

تضمن البصمة الالكترونية Digest Message تكامل الرسالة وعدم تعديلها. لكن البصمة الكترونية لا تحدد هوية مرسل الرسالة. تسمى البصمة الالكترونية التي تنتج عن استخدام تابع تهشير "كود التقاط التعديل" Modification detection code (MDC) وهو يساعد على تحديد فيما إذا عدلت الرسالة أم لا. ما نحتاج إليه من أجل تحديد الهوية ما يسمى "كود هوية الرسالة" Meassge Authentication Code (MAC).

كود التقاط التعديل Modification Detection Code (MDC)

يوضح الشكل التالي كيفية استخدام هذا الكود:

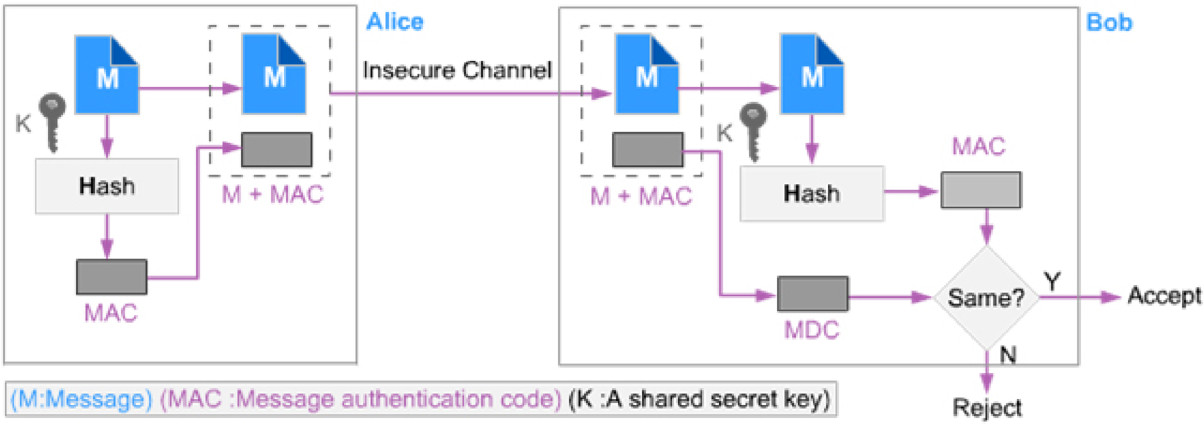


(M:Message) (Hash :Cryptographic hash function) (MDC: Modification detection code)

يجري تبادل الكود MDC من خلال قناة آمنة وعند وصول الرسالة إلى الطرف الثاني يعاد حساب تابع التهشير لانتاج كود MDC جديد ومقارنته مع الكود الأصلي المرسل مسبقاً وتحديد فيما إذا خضعت الرسالة إلى التعديل أم لا.

كود هوية الرسالة Message Authentication Code (MAC)

للتأكد من تكامل الرسالة وتحديد هوية المرسل، نحتاج إلى الاستعاضة عن كود التقاط التعديل بكود آخر هو كود هوية الرسالة MAC. يتضمن هذا الكود الجديد مفتاح سري بين طرفي التراسل وهو غير معروف من قبل أطراف أخرى. يشرح الشكل التالي الفكرة التي نتحدث عنها:



يجري انتاج الكود **MAC** بعد تطبيق تابع التهشير على الرسالة والمفتاح معاً: **h(K|M)**.

ثم ترسل الرسالة والكود **MAC** المرافق لها من خلال قناة الاتصال غير الآمنة إلى الطرف الآخر.

بعد تسلم الرسالة من قبل الطرف الآخر، يقوم هذا الطرف بإعادة حساب الكود **MAC** للرسالة بعد فصلها عن الكود **MAC** الأصلي، ثم يجري مقارنة الكودين للتحقق من تكامل الرسالة وللتأكد من أنها مرسلة من قبل الطرف الأول الذي يعرف دون غيره المفتاح السري.

الفصل السابع

توابع التهشير Hash Functions

الأهداف Objectives

يهدف هذا الفصل إلى التعريف بما يلي:

- الفكرة الكامنة وراء توابع التهشير والحاجة إليها ضمن عالم التشفير.
- تعريف توابع التهشير التكرارية.
- التمييز بين فئتين من توابع التهشير: تلك التي تستخدم تابع ضغط من الصفر وتلك التي تستخدم مشفر كتلي كتابع ضغط.
- عرض بنية الخوارزمية **SHA-512** كمثال عن توابع التهشير التي تستخدم تابع ضغط من الصفر.

1. مقدمة Introducation

كما تحدثنا سابقاً، تقوم توابع التهشير بأخذ رسالة من طول متغير لتقوم ببناء بصمة الكترونية لها من طول ثابت. سنبين في هذ الفصل تفاصيل خوارزميتين للتهشير، هما:

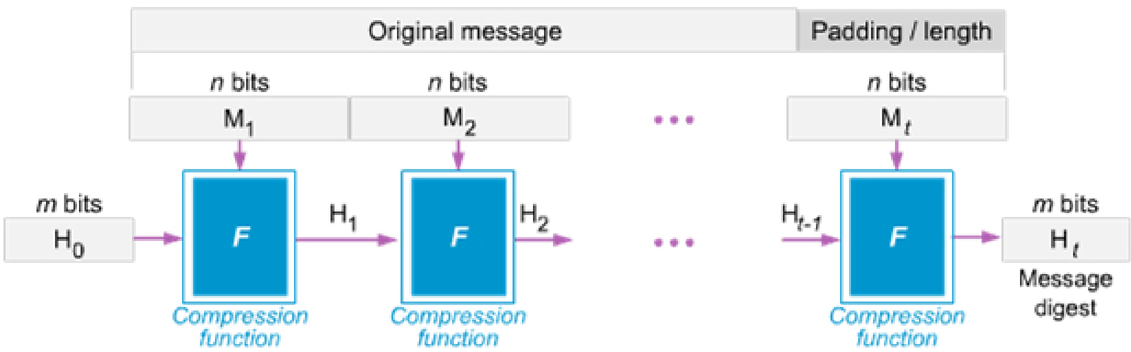
SHA-512 و Whirlpool.

قبل البدء بذلك نحتاج إلى توضيح بعض الأفكار العامة التي يمكن تطبيقها عند التعامل مع توابع التهشير.

تابع التهشير التكراري Iterated Hash Function

يمكن إنشاء بصمة الكترونية من طول ثابت لرسالة من طول متغير بصورة تكرارية وذلك بتكرار استخدام تابع تهشير ذو دخل من حجم ثابت العدد الكافي من المرات لتغطية كامل الرسالة. نسمي التابع ذو الدخل الثابت بتابع الـضغط **Compression Function** لأنه يقوم بضغط سلسلة من **n** بت إلى سلسلة من **m** بت حيث لدينا **m** أصغر بكثير من **n**.

يبين الشكل التالي الشكل العام لهذا النوع من توابـع التهشير والمسمى **Merkle-Damgard Scheme**. يتضمن هذا المخطط تابع ضغط مقاوم للتصادم **Collision Resistance**.



يوضح هذا المخطط أنه جرى تقسيم الدخل إلى عدة كتل من حجم مساوي لـ n بت، حيث إضيف إلى الكتلة الأخيرة بعض معطيات الحشو لاستكمال طولها إلى n بت إذا كان ذلك ضرورياً.

تتحول الرسالة بالنتيجة إلى t كتلة $M_1, M_2, \ldots, M_t$، كل منها من حجم مـساوي لـ n بـت. نسمي البصمة الناتجة من خلال t تكرار بـ $H_1, H_2, \ldots, H_t$. قبل البدء تأخـذ البـصمة H_0 قيمة ثابتة وتسمى القيمة IV (Initial value).

في كل مرحلة من التهشير لدينا $H_i = f(H_{i-1}, M_i)$ حيث أن f هو تابع الضغط. القيمة H_t هي نتيجة تابع التهشير للرسالة الأصلية أي: h(M).

فئتان من توابع الضغط Two Groups of Compression Functions

نحتاج إلى تصميم تابع ضغط مقاوم للتصادم لإقحامه ضمن المخطط السابق. يوجد طريقتين لتحقيق ذلك:

الطريقة الأولى: تصميم تابع تهشير من الصفر.

الطريقة الثانية: استخدام مشفرات كتلية.

الطريقة الأولى: تصميم تابع تهشير من الصفر

من بين هذه التوابع لدينا التـابعين Message Digest (MD) و Secure Hash Algorithm (SHA).

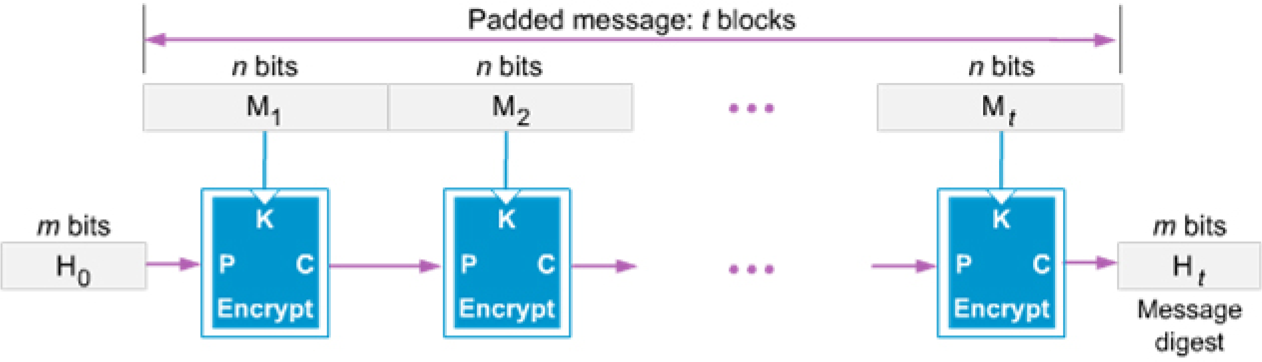
- **التابع Message Digest (MD):** جرى تصميم هذا التابع من قبل Ron Rivest. ويرمز له وفقاً لأحد المسميات التالية: MD2, MD4, MD5. تتميز النسخة الأخيرة MD5 في كونها أقوى من سابقاتها، فهي تقوم بتقسيم الرسالة إلى كتل بحجم مساوي لـ 512 بت وتنشئ بصمة بحجم 128 بت. يعتبر حجم البصمة هذا صغير جداً ليقاوم التصادم.
- **التابع Secure Hash Algorithm (SHA):** وهو تابع معياري مصمم من قبل المعهد NIST. يعتمد هذا المعيار على MD5. جرى تطويره عام 1995 تحت المسمى SHA-1. تمت مراجعته عدة مرات وانتج ذلك النسخ التالية: SHA-224, SHA-256, SHA-384, SHA-512. تتمتع كافة هذه النسخ بنفس البنية وسوف نتحدث بالتفصيل عن التابع SHA-512 فيما بعد.

الطريقة الثانية: استخدام مشفرات كتلية

يمكن استخدام مشفر كتلي بمفتاح متناظر كتابع ضغط. تم التفكير بذلك لأنه يتوفر لدينا العديد من خوارزميات التشفير الكتلي بمفتاح متناظر (مثل: **DES** و **AES**) ولا داع للتفكير لبناء تابع ضغط جديد.

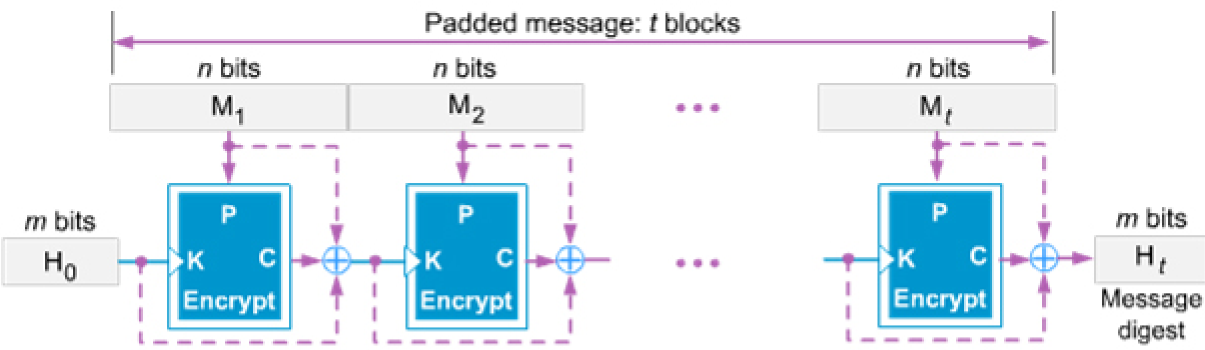
يتوفر عدة مخططات لبناء توابع التهشير التي تستخدم مشفرات كتلية كتوابع ضغط، والتي نذكر منها:

مخطط رابين **Rabin Scheme** المبين في الشكل التالي:



يشبه هذا المخطط مخطط **Merkle-Damgrad** حيث تم الاستعاضة عن تابع الضغط بمشفر كتلي حيث تستخدم كتل الرسالة كمفاتيح تشفير والنص المشفر الناتج هو بصمة الرسالة. لاحظ أن حجم البصمة هو حجم الكتل التي يتعامل معها المشفر. يعاني هذا المخطط من الهجوم "المقابلة من الوسط" **meet in middle**.

مخطط **Miyaguchi-Preneel** المبين في الشكل التالي:



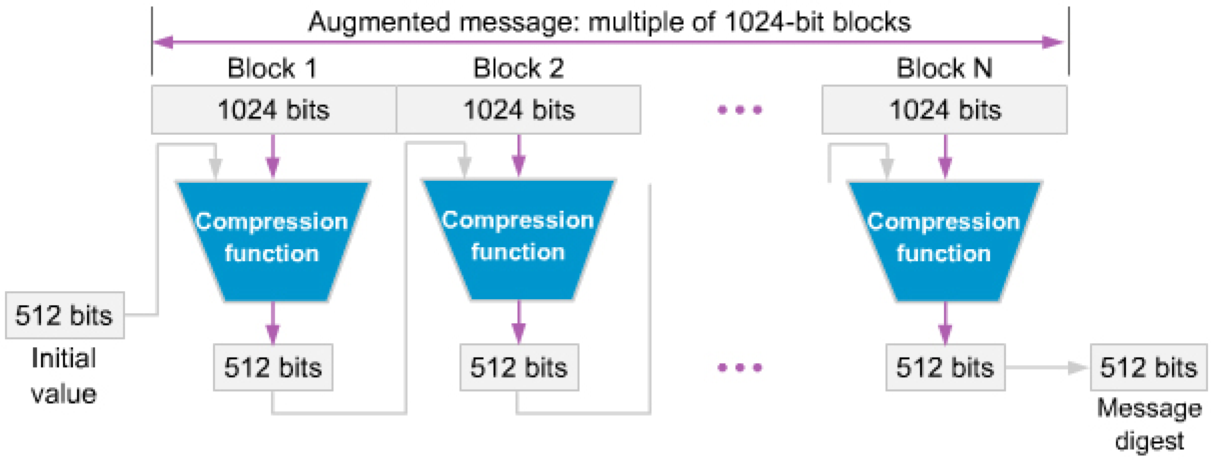
يستخدم هذا المخطط مع الخوارزمية **Whirlpool**. هنا يجري تطبيق التابع **XOR** على خرج كل مرحلة تشفير مع خرج المرحلة السابقة مع الرسالة. يقاوم هذا المخطط الهجوم "المقابلة من الوسط" **meet in middle** الذي يعاني منه المخطط السابق.

2. SHA-512

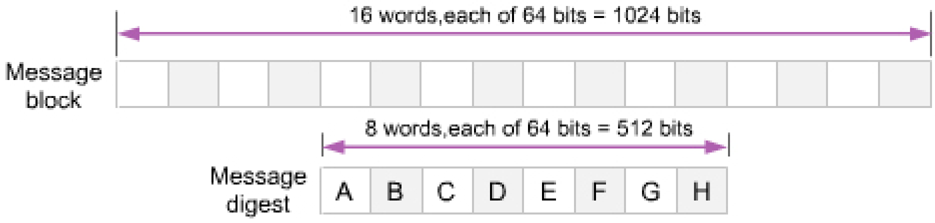
يعتمد على مخطط **Merkle-Damgard** و**MD5**. ويعتبر معيار **NIST**. وهو يستند في عمله على الأفكار التالية:

- تطبيق عدد كبير من الجولات (**80** جولة) للتمويه بأكبر قدر.
- يستخدم بكثرة توابع غير قابلة للعكس.
- جرى تصميمه لينجز عمله بسرعة وخاصة عند تحقيقه على مستوى البنى الصلبة.

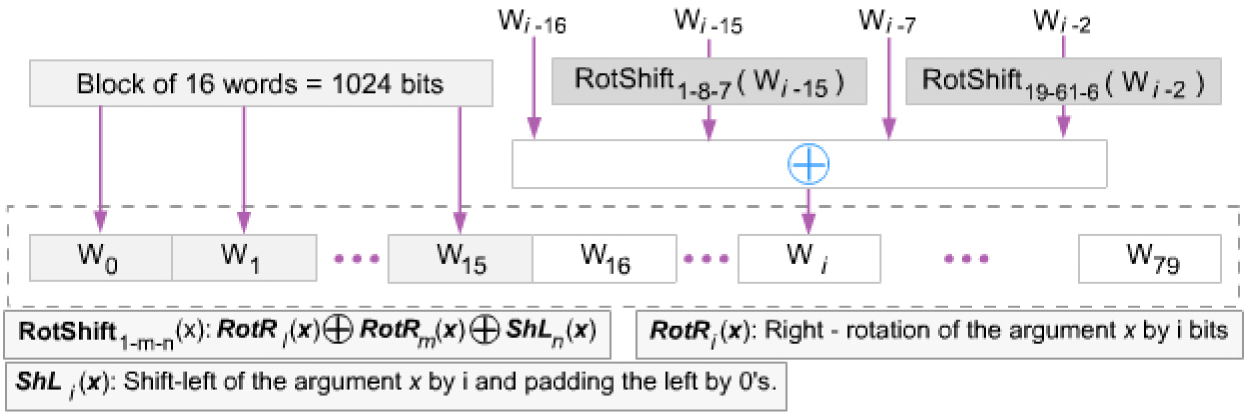
يبين الشكل التالي كيفية إنشاء بصمة لرسالة من طول مساوي لـ **512** بت:



يعمل **SHA-512** على مستوى الكلمات بحجوم **64** بت. يعني ذلك أنه بعد حشو الرسالة وإضافة طول الرسالة الحقيقي إلى نهاية الرسالة، يجري تقسيمها إلى كتل من **16** كلمة. يجري تشكل بصمة الرسالة على شكل كلمات بطول **64** بت ولكن تتألف البصمة النهائية من **8** كلمات فقط، وهي: الكلمات **A, B, C, D, E, F, H** كما هو مبين في الشكل التالي:

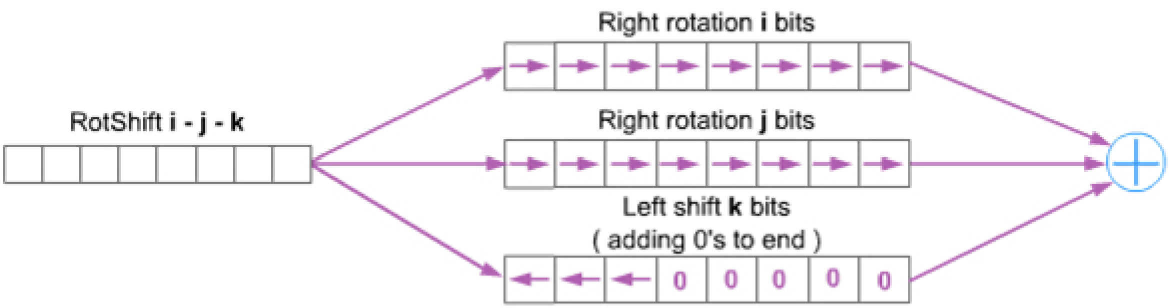


على اعتبار أنه لدينا كتل بحجم **16** كلمة فإنه يجب توسيعها لتتضمن **80** كلمة. تتم هذه العملية كما هو مبين في الشكل التالي:



يجري بناء الكلمات الجديدة اعتماداً على أربع كلمات سابقة حيث يطبق عليها التابع **XOR** لانتاج الكلمة المطلوبة.

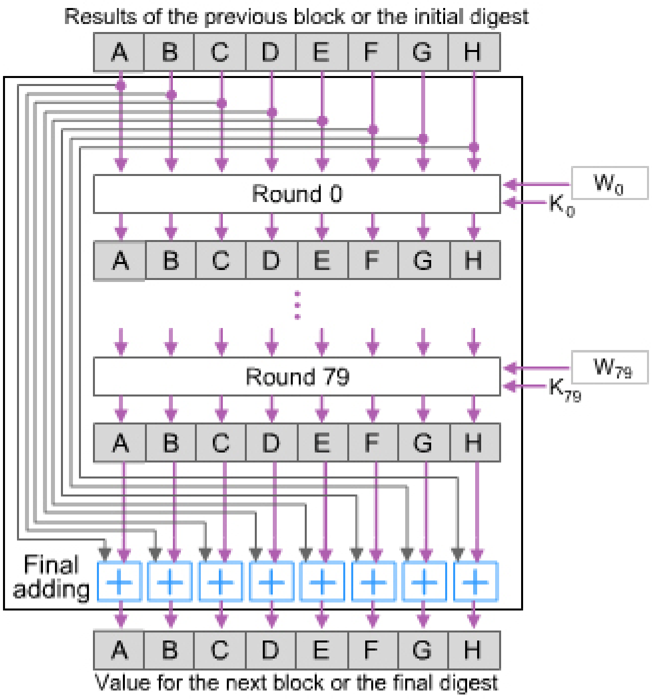
أثناء تنفيذ ذلك يتم تطبيق عمليات دوران وإزاحة لزيادة التمويه باستخدام تابع الإزاحة التالي:



تستخدم الخوارزمية ثمانية قيم ابتدائية والثوابت التي نسميها من A_0 إلى H_0 . يبين الجدول التالي قيم هذه الثوابت:

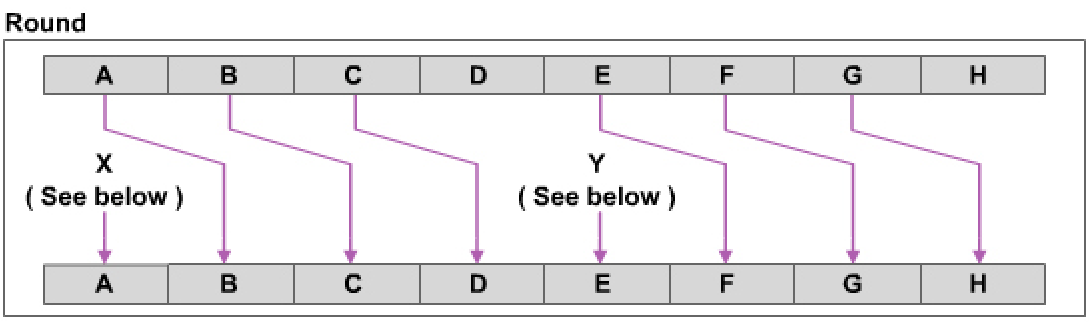
A	6A09E667F3BCC908
B	BB67AE8584CAA73B
C	3C6EF372EF94F828
D	A54FE53A5F1D36F1
E	510E527FADE682D1
F	9B05688C2B3E6C1F
G	1F83D9ABFB41BD6B
H	5BE0CD19137E2179

جرى حساب القيم الثابتة السابقة اعتماداً على الأعداد الأولية الثمانية الأولى: 2, 3, 5, 7, 11, 13, 17, 19 حيث نأخذ قيمة الجزء الكسر للجذر المربع للعدد الأول الموافق لكل ثابت ذلك بعد تحويله إلى التمثيل الثاني والاحتفاظ بالخانات 64 بت.

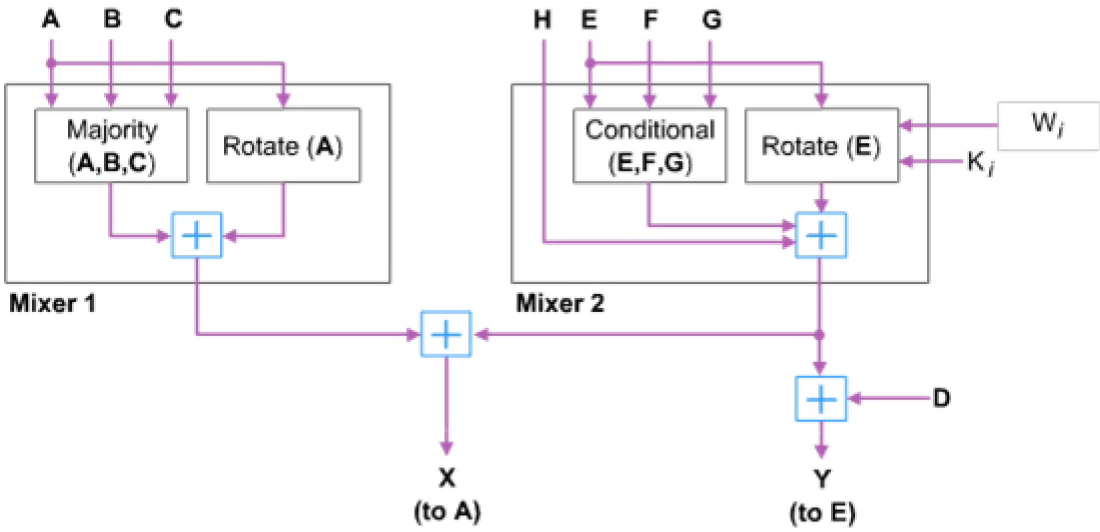


أما توابع الضغط فهي مكونة من 80 جولة حيث تقوم كل منها بحساب بصمة وسيطة ويتم في المرحلة الأخيرة حساب مجموع بصمة الجولة الأولى مع بصمة الجولة الأخيرة ($mod\ \ 2^{64}$) كما هو مبين في الشكل التالي:

يجري ضمن كل جولة من جولات تابع الضغط (رقم i مثلاً) استخدام البصمة السابقة والكلمة W_i والمفتاح K_i. يجري حساب المفاتيح K_i اعتماداً على الأعداد الأولية الـ 80 الأولى، حيث نأخذ قيمة الجزء الكسر للجذر المربع للعدد الأول الموافق لكل مفتاح ذلك بعد تحويله إلى التمثيل الثاني والاحتفاظ بالخانات 64 بت. أما كل جولة فتتكون من المكونات المبينة في الشكل التالي:



أما X و Y فيجري حسابهما كما يلي:



تتم عملية التدوير بموجب التابع **Rotate(x)** الذي يقوم بتنفيذ مايلي:

$$\text{Rotate}(x) = RotR_{28}(x) \oplus RotR_{34}(x) \oplus RotR_{39}(x)$$

وهي عمليات إزاحة إلى اليمين بمقدار **28** و**34** و**39** بت. ثم يتم الجمع بينها باستخدام التابع **XOR**.

أما التابع **Majority(A,B,C)** فيعمل كما يلي:

I^th bit of result = 1 if at least 2 of i^th bits of A, B, C = 1

0 oherwise

مثال1:

A = 11001010
B = 01101001
C = 10011101

Majority= 11001001

وبالنسبة للتابع **Conditional(E,F,G)** فيعمل وفقاً لما يلي:

I^th bit of result = i^th bit of F if i^th bit of E = 1

= i^th bit of G oherwise

الفصل الثامن

التوقيع الإلكتروني Digital Signature

الأهداف Objectives

يهدف هذا الفصل إلى التعريف بما يلي:

- ماهية التوقيع الالكتروني واستخداماته.
- تعريف الخدمات الأمنية التي يوفرها التوقيع الالكتروني.
- مناقشة بعض مخططات التوقيع الالكتروني، مثل: **RSA** وغيرها.

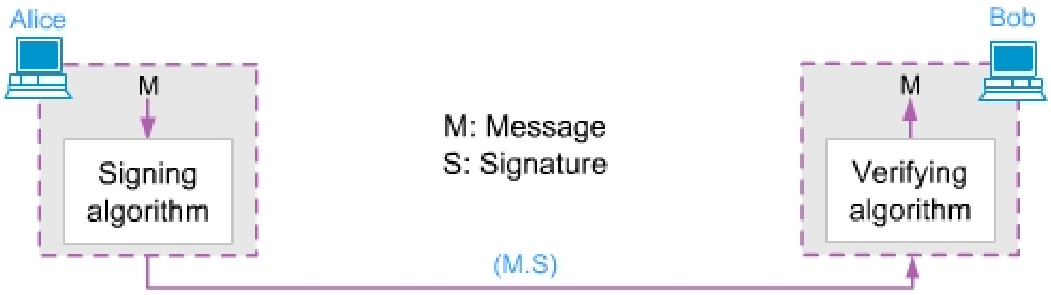
1. مقارنة Comparison

نعرض فيما يلي الفروق بين التوقيع العادي المتعارف عليه والتوقيع الالكتروني:

- **التواجد Inclusion**: يتواجد التوقيع العادي على الوثائق وهو يكون جزءاً لا يتجزء من هذه الوثائق، مثلاً، عند كتابة شيك، فإن التوقيع يكون متواجداً عليه ولا يمكن فصله عنه. ولكن عند توقيع وثيقة ما رقمياً فإنه يمكن أن نرسل التوقيع على شكل وثيقة منفصلة، حيث يستقبل المتلقي نصين: الوثيقة وتوقيعها الالكتروني. يقوم بعدها المتلقي بالتحقق من صحة توقيع المرسل، فإذا كان التوقيع صحيحاً، تقبل الرسالة وإلا فإنها سترفض.
- **طريقة التحقق Verification Method**: مع التواقيع العادية، يجري التحقق من صحة التوقيع بمقارنة التوقيع مع نسخة التوقيع المتوفرة ضمن سجل التواقيع. أما بالنسبة للتوقيع الالكتروني، يتلقى المتلقي الرسالة والتوقيع. لا يوجد لدى المتلقي أية نسخة سابقة عن توقيع المرسل، وإنما يجب عليه تطبيق تقنية للتحقق على الرسالة والتوقيع للتحقق من هوية المرسل.
- **العلاقة Relationship**: بالنسبة للتوقيع العادي، فلدينا علاقة واحد لمتعدد بين التوقيع والوثائق. يمكن لشخص ما أن يستخدم توقيعه نفسه لتوقيع عدة وثائق. في حين بالنسبة للتوقيع الالكتروني، فيوجد علاقة واحد لواحد بين التوقيع والرسائل. لكل رسالة لديها توقيعها الخاص بها.
- **التكرار Duplicity**: مع التواقيع العادية، يجري التفريق بين توقيع النسخ الأصلية والنسخ المكررة. أما بالنسبة للتوقيع الالكتروني، فإنه لا يوجد أي تفريق إلا إذا تعلق الأمر بالزمن.

2. آلية العمل Process

يوضح الشكل التالي آلية عمل التوقيع الالكتروني:



يقوم المرسل باستخدام خوارزمية توقيع **Signing algorithm** لتوقيع رسالته. ثم يجري نقل الرسالة وتوقيعها. يطبق المتلقي عليهما خوارزمية التحقق **Verifying algorithm**. إذا كانت النتيجة صحيحة فتقبل الرسالة وإلا فترفض.

الحاجة لمفاتيح Need for Keys

يعتبر التوقيع العادي كما لو أنه مفتاح خاص بحوزة صاحب الوثيقة الموقعة. من حيث المبدأ لا يمتلك أحد آخر لستخدام هذا التوقيع لتوقيع الوثائق. تشبه نسخة التوقيع الموجودة في سجل التواقيع المفتاح العام بحيث لأي كان الاطلاع على هذا السجل للتأكد من صحة التوقيع ومقارنته مع التوقيع الموجود على الوثائق.

بالنسبة للتوقيع الالكتروني، يقوم الموقع باستخدام مفتاحه الخاص ويطبقه باستخدام خوارزمية التوقيع لتوقيع الوثيقة. يقوم الشخص الذي يسعى للتحقق من صحة التوقيع باستخدام المفتاح العام للموقع ويطبقه باستخدام خوارزمية التحقق.

يوضح الشكل التالي الفكرة التي قمنا بشرحها:

السؤال الذي يطرح نفسه، هل يمكن استخدام مفتاح سري (متناظر) للتوقيع وللتحقق من صحة التوقيع؟ الجواب هو لا وذلك لعدة أسباب:

أولاً: إن المفتاح السري معروف من قبل طرفين فقط. بالتالي، إذا احتاج شخص إلى توقيع وثيقة أخرى وارسالها إلى شخص ما فإنه يجب استخدام مفتاح سري آخر مختلف عن مفتاح الرسالة السابقة.

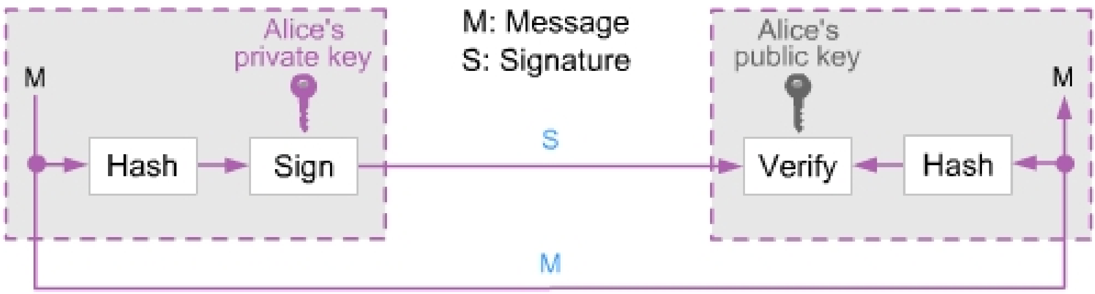
ثانياً: لإنشاء مفتاح سري فإننا نحتاج إلى تنفيذ عملية تحديد الهوية التي تتطلب استخدام التوقيع الالكتروني، أي أننا ندخل بالتالي في حلقة مفرغة.

ثالثاً: يمكن لشخص أن يستخدم المفتاح السري بينه وبين شخص أول لتوقيع رسالة ليرسلها إلى شخص ثالث ويدعي كما لو أنها قادمة من الشخص الأول. نصل بالتالي أننا من أجل التوقيع الالكتروني، نحتاج إلى نظام معتمد على التعامل مع مفاتيح عامة.

توقيع البصمة Signing the Digest

إن عملية إنشاء توقيع الكتروني لكامل محتوى رسالة هي عملية مكلفة ومساوية في كلفتها كلفة تشفير كامل محتوى الرسالة باستخدام خوارزمية مفتاح عام. كيف يجب علينا تجاوز هذه المعضلة؟

لحل هذه المشكلة، يمكن لنا استخدام بصمة الرسالة بدلاً من كامل محتوى الرسالة. ويمكن لنا إنشاء توقيع من البصمة التي هي أصغر بكثير من الرسالة. يقوم المتلقي بإنشاء نفس البصمة للرسالة المستقبلة، ثم يقارن بين التواقيع مستخدماً بصمة الرسالة في هذه العملية. يوضح الشكل التالي هذه الفكرة:



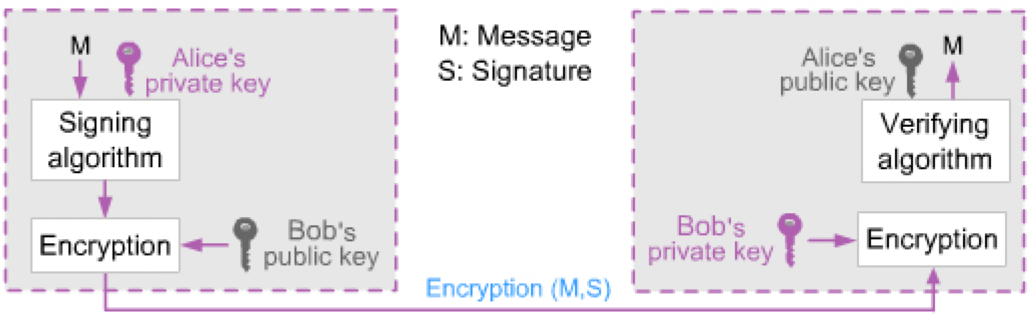
3. الخدمات الأمنية Security Services

تحدثنا في الفصل الأول عن عدة خدمات أمنية ونذكر من بينها الخدمات التالية: سرية الرسائل **Message confidentiality**، تحديد هوية المرسل **Message authentication**، تكامل الرسائل **Message integrity**، وعدم القدرة على الانكار **Nonrepudiation**.

استناداً إلى ما تم شرحه فيما سبق، يستطيع التوقيع الالكتروني توفير على الأقل الخدمات الأمنية الثلاث الأخيرة من بين الخدمات المذكورة أعلاه:

- يوفر التوقيع الالكتروني إمكانية تحديد هوية المرسل بسبب استخدام المفتاح العام المرتبط به في عملية التحقق.
- يوفر التوقيع الالكتروني إمكانية تكامل الرسالة لأن أي تعديل فيها لن يعطي نفس التوقيع الالكتروني للرسالة.
- يمكن توفير خدمة "عدم الانكار" مروراً بطرف ثالث موثوق (مركز ثقة **Trusted Center**).

أما بالنسبة للحاجة إلى التشفير وفك التشفير فيجب إضافتها كطبقة إضافية كما هو مبين في الشكل التالي:



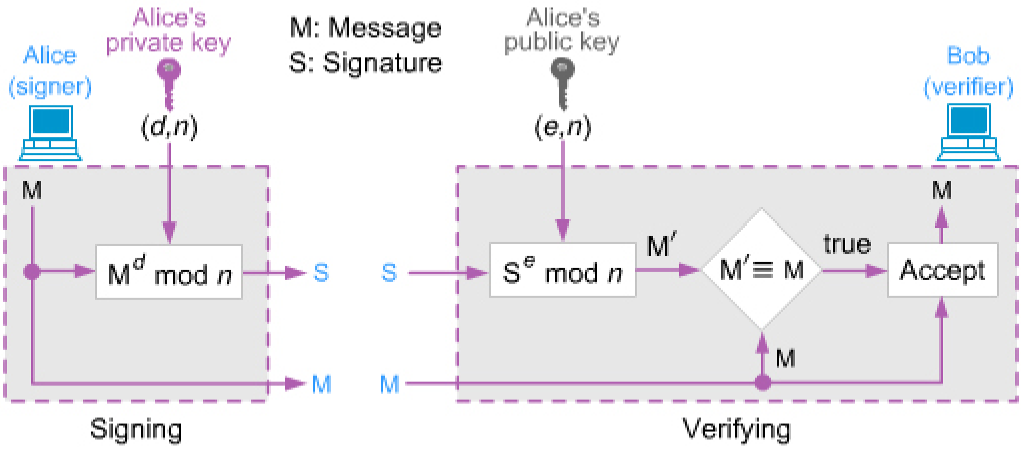
4. مخطط التوقيع الالكتروني RSA Digital Signature Scheme RSA

شرحنا فيما سبق كيف يمكن استخدام نظام التشفير **RSA** من أجل توفير الخصوصية والسرية. يمكن أيضاً استخدام **RSA** من أجل توقيع الرسائل والتحقق منها. يقوم مخطط التوقيع الالكتروني **RSA** بتغيير دور المفتاحين العام والخاص:

- يجري استخدام المفتاحين العام والخاص للمرسل وليس المستقبل للرسالة.
- يستخدم المرسل المفتاح الخاص به للتوقيع على الرسالة. أما المستقبل فيستخدم مفتاح

المرسل العام للتحقق من التوقيع.

لتوليد المفتاح المعتمد عليها في مخطط العمل المذكور فيما سبق للتوقيع الالكتروني، نتبع نفس مخطط العمل المرتبط بالنظام **RSA**. يوضح الشكل التالي مخطط العمل الذي نتحدث عنه:



كما يبين هذا الشكل كيفية استخدام المفاتيح لتوقيع الرسائل والتحقق من ذلك. لاحظ أنه في عملية التوقيع جرى استخدام المفتاح الخاص للمرسل. أما المفتاح العام للمرسل فجرى الاعتماد عليه في عملية التحقق.

الفصل التاسع
إجرائيات تحديد الهوية

Authentication Procedures

الأهداف Objectives

يهدف هذا الفصل إلى التعريف بما يلي:

- التمييز بين تحديد هوية مرسل رسالة **Message Authentication** وتحديد هوية كيان **Entity Authentication**.
- الحديث عن بعض أساليب تحديد هوية الكيانات باستخدام كلمات السر **Passwords**.
- الحديث عن بروتوكولات التجاوب مع التحدي **Challenge-response protocols** لتحديد هوية الكيانات.

1. مقدمة Introduction

تعتبر عملية تحديد هوية الكيانات **Entity authentication** واحدة من التقنيات المصممة للسماح لطرف ما بتحديد هوية طرف آخر. يمكن لكيان أن يكون شخص أو إجراء أو زبون أو مخدم. نستخدم مصطلح المدّعي **Claimant** للحديث عن الكيان الذي نحتاج لتحديد هويته. ونسمي الطرف الذي دعا لتحديد هوية المدعي بالمتحقق **Verifier**.

مثال: عندما يحاول **Bob** تحديد هوية **Alice**، فإن **Alice** هو المدعي. أما **Bob** فهو المتحقق.

هوية المرسل وهوية الكيان Message Authentication Versus Entity Authentication

يوجد فرقان أساسيان بين عملية تحديد هوية مرسل رسالة التي تحدثنا عنها في الفصل السابق وعملية تحديد هوية كيان التي سنتحدث عنها في هذا الفصل:

1. لا يمكن تنفيذ عملية تحديد هوية مرسل رسالة بالزمن الحقيقي. أما بالنسبة لعملية تحديد هوية كيان فيمكن ذلك.
2. يجري تنفيذ عملية تحديد هوية مرسل رسالة محددة ويجب إعادة العملية من أجل رسالة أخرى. أما بالنسبة لعملية تحديد هوية كيان فإن نتائج هذه العملية تبقى صالحة طيلة فترة جلسة الاتصال.

فئات التحقق Verification Categories

يمكن أن تتم عملية التحقق من هوية كيان من ثلاثة أنواع من الشواهد، وهي:

- بواسطة **شيء تعرفه Something known**: وهو عبارة عن سر يعلمه المدعي فقط

ويمكن للمتحقق أن يتأكد منه. أمثلة: كلمة السر، رقم التعريف الشخصي **PIN**، مفتاح سري، أو مفتاح خاص.

- بواسطة **شيء تملكه Something possessed**: وهو عبارة عن شيء يمكن أن يبرهن عن هوية المدعي. أمثلة: جواز سفر، شهادة سواقة، بطاقة هوية، بطاقة دفع الكتروني وبطاقة ذكية.
- بواسطة **شيء موروث Something inherent**: وهو عبارة عن صفة وراثية مميزة للمدعي. أمثلة: التوقيع العادي، بصمات الأصابع، الصوت، قسمات الوجه، وبصمة اليد.

2. كلمات السر Passwords

وهي من أقدم طرق تحديد هوية الكيانات، حيث ترمز كلمة السر لشيء يعرفه المدعي. تستخدم كلمات السر عندما يرغب مستخدم في الوصول إلى نظام للتعامل مع مصادره. يمتلك كل مستخدم معرف عام وعلني وكلمة سر خاصة به. يمكن تقسيم مخططات تحديد الهوية باستخدام كلمات السر إلى فئتين، هما:

1. فئة كلمات السر الثابتة **Fixed passwords**.
2. فئة كلمات السر لمرة واحدة **One-time passwords**.

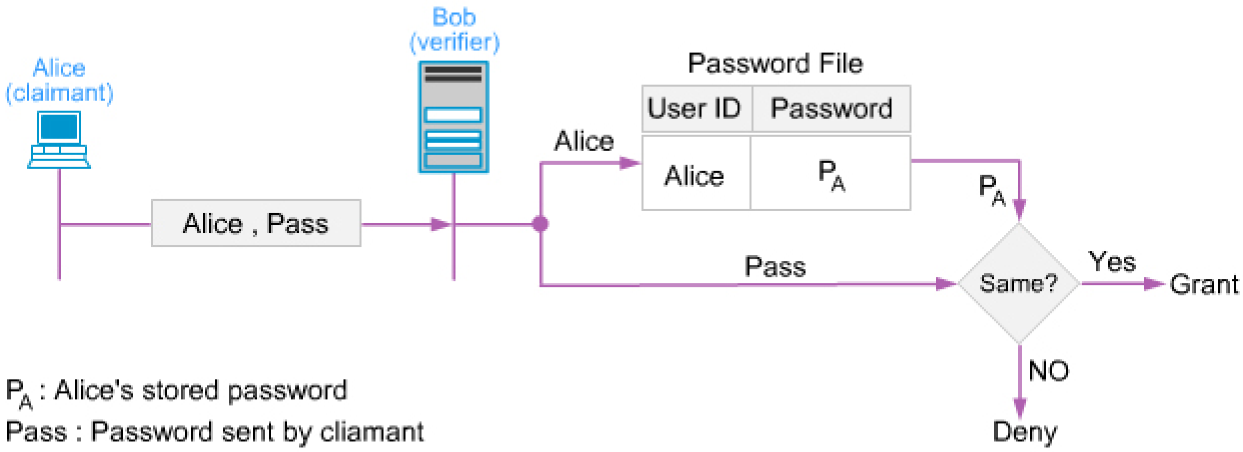
كلمات السر الثابتة Fixed Passwords

وهي كلمات السر المستخدمة مرات ومرات مع كل محاولة وصول. يتوفر لدينا عدة مخططات لبنائها وتحقيقها:

1. ملف كلمات السر.
2. تهشير كلمات السر.
3. تمليح كلمات السر.

ملف كلمات السر Password File

يحتفظ النظام بكلمات السر ضمن ملف (أو جدول). وبالتالي عند محاولة الوصول إلى مصادر هذا النظام فإن المستخدم يرسل اسمه وكلمة سره إلى النظام دون تشفير. يقوم النظام بعدها بالبحث ضمن جدول كلمات السر ويطابق بين كلمة السر المخزنة لديه وتلك المرسلة، كما هو مبين في الشكل التالي:

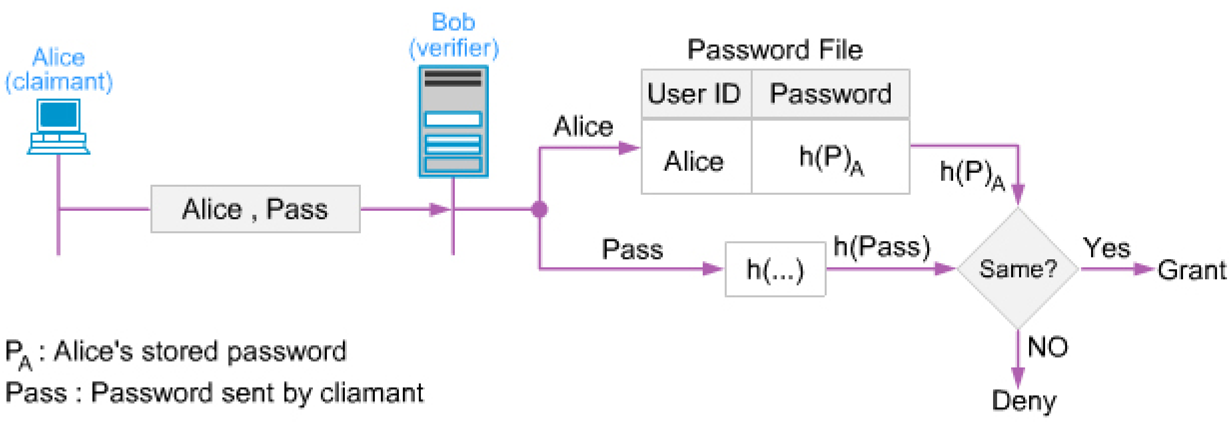


تعاني هذه الطريقة من الإشكالات التالية:

1. يمكن مثلاً التنصت على الاتصل بين **Alice** و**Bob** ونسخ كلمة السر المرسلة.
2. يمكن سرقة كلمة السر المسجلة فيزيائياً في مكان ما، أو المرتبطة بشيء معروف عن **Alice**.
3. يمكن محاولة الوصول إلى ملف كلمات السر ضمن المخدم.
4. حزر كلمة السر من خلال محاولة توليدها ومحاولة الوصول إلى المخدم بواسطة الكلمات المولدة.

تهشير كلمات السر Hashing the password

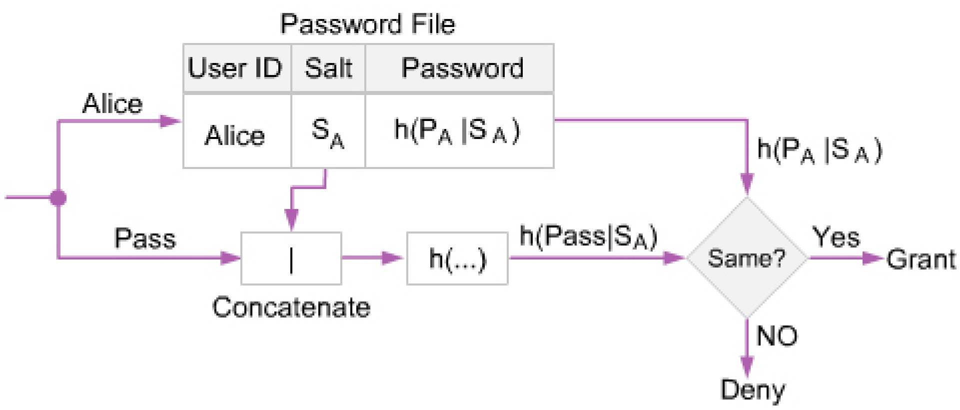
لضمان سرية كلمات السر يمكن تخزينها ضمن جدول أو ملف كلمات السر بعد تطبيق تابع تهشير عليها. تتم عملية تحديد الهوية باستخدام كلمات السر المهشرة بشكل مشابه لسيناريو العمل السابق مع الأخذ بعين الاعتبار لضرورة تطبيق تابع التهشير على كلمة السر المقدمة من قبل المدعي ومقارنة النتيجة مع التهشير المخزن مسبقاً ضمن ملف كلمات السر، كما هو مبين في الشكل التالي:



تعاني هذه الطريقة من أن كلمات السر يمكن أن لا تقاوم هجوم من النوع الذي يستخدم قاموساً لتجريب عدد كبير من الكلمات يعود سبب ذلك إلى صغر طول كلمة السر في بعض الأحيان مما يسهل عملية كشفها.

تمليح كلمات السر Salting the passwords

تقدم هذه الطريقة حلاً لمشكلة الهجوم باستخدام قاموس لكشف كلمة السر. يمكن التقليل من ذلك بزيادة طول كلمة السر وذلك بإضافة سلسلة عشوائية من الحروف إليها. تسمى هذه السلسلة بالملح **Salt**. يجري بعدها تطبيق تابع تهشير على الكلمة الناتجة عن دمج كلمة السر والملح المضاف إليها كما هو مبين في الشكل التالي:



كلمات السر لمرة واحدة One-time Passwords

تعتمد فكرة كلمات السر من هذا النوع على استخدامها لمرة واحدة فقط. يوجد ثلاثة طرق لتحقيق ذلك، وهي:

- استخدام كلمة من بين مجموعة من كلمات السر المتفق عليها بين المستخدم والنظام. يجري تخزين كلمات السر ضمن النظام ضمن قائمة ويتم استخدام كل كلمة مرة واحدة فقط. تعاني هذه الطريقة من أنه يجب على المستخدم والنظام تخزين قائمة طويلة من كلمات السر. يضاف إلى ذلك أنه في كل مرة يجري التعامل فيها مع كلمة سر يجب البحث عنها ضمن القائمة والتأكد من وجودها، وهذه العملية هي عملية بطيئة.
- يتم التوافق بين المستخدم والنظام على تغيير كلمة السر تسلسلياً. يتفق الطرفان على كلمة سر ابتدائية P_1 وهي صالحة للوصول إلى النظام للمرة الأولى فقط. ثم أثناء العمل على النظام للمرة الأولى، يتم تعديل كلمة السر لتصبح P_2 ، ثم تشفيرها باستخدام كلمة السر السابقة P_1 كمفتاح. وتعامل كلمة السر P_2 بنفس الطريقة وهكذا دواليك.

- يتم التوافق بين المستخدم والنظام على إنشاء كلمات سر يمكن تعديلها تسلسلياً باستخدام تابع تهشير. يتم ذلك وفقاً لخوارزمية عمل طورها **Leslie Lamport**. تعمل هذه الخوارزمية كما يلي:
  1. نبدأ بكلمة السر الابتدائية P_0 .
  2. لدينا عداد **n** مزود بقيمة ابتدائية **n0**.
  3. يجري انقاص قيمة العداد تدريجياً من قبل المدعي والمتحقق بعد كل عملية وصول.
  4. كلمة السر الحالية هي كلمة السر الابتدائية بعد تهشيرها **n** مرة.
  5. $P_i = h^{n0-i}(P_0)$.

3. برتوكولات التجاوب مع التحدي Challenge-Response

تعتمد أساليب تحديد الهوية باستخدام كلمات السر على فكرة أن المدعي قادر على أن يبرهن أنه يعرف كلمة السر. وعلى اعتبار أنه يقدم هذا السر ويكشفه فسوف يكون عرضة للالتقاط من قبل الآخرين. أما بالنسبة لبرتوكولات التجاوب مع التحدي فإن المدعي يقدم البرهان على أنه يعرف السر دون أن يرسله. مع هذه البرتوكولات، يتغير التحدي مع تغير الزمن فيمكن أن يكون مثلاً عدداً عشوائياً أو طابع زمني مرسل من قبل المتحقق. يقوم المدعي بتطبيق تابع على التحدي ويرسل جواباً إلى المتحقق. يمكن للجواب المرسل أن يبرهن أن المدعي يعرف السر.

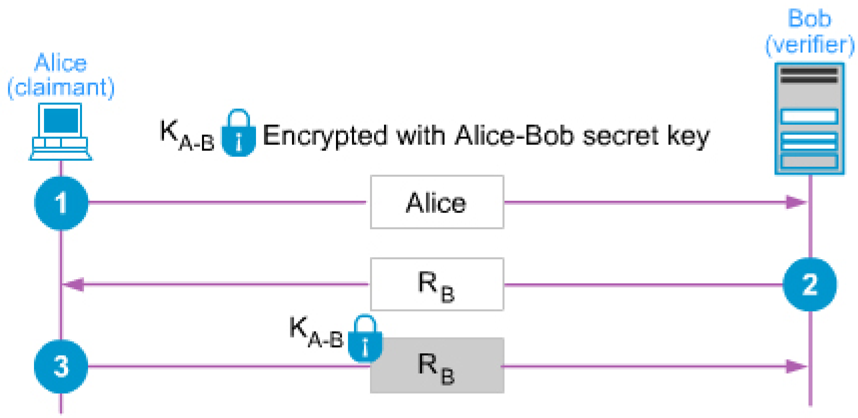
يتوفر عدة طرق لتحقيق برتوكولات التجاوب مع التحدي والتي نذكر منها:

- **استخدام مشفر مزود بمفتاح متناظر:**

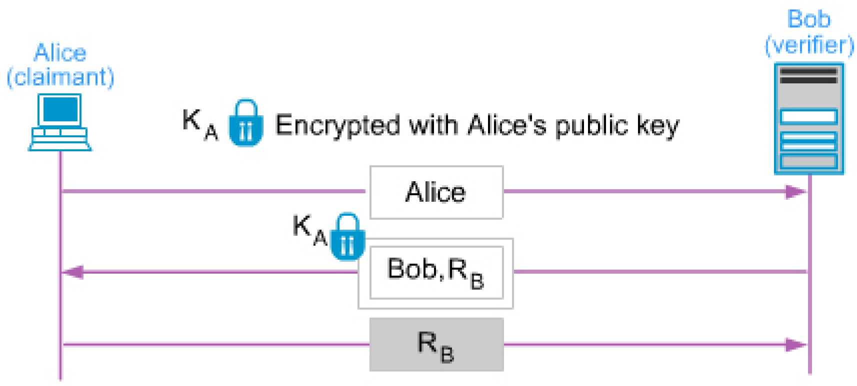
تعتمد فكرة هذه الطريقة على تبادل عدة رسائل تكون كافية للبرهان على أن المدعي يعرف السر . لا تشكل الرسالة الأولى جزءاً من بوتوكول التجاوب مع التحدي. وإنما تقوم بإخبار المتحقق أن المدعي يرغب بالتحدي.

يرسل المتحقق بعدها التحدي للمدعي الذي يقوم بدوره بإعادته إلى المتحقق بعد بواسطة السر المشترك بينهما (وهو المفتاح المتناظر المشترك) كما هو مبين في الشكل.

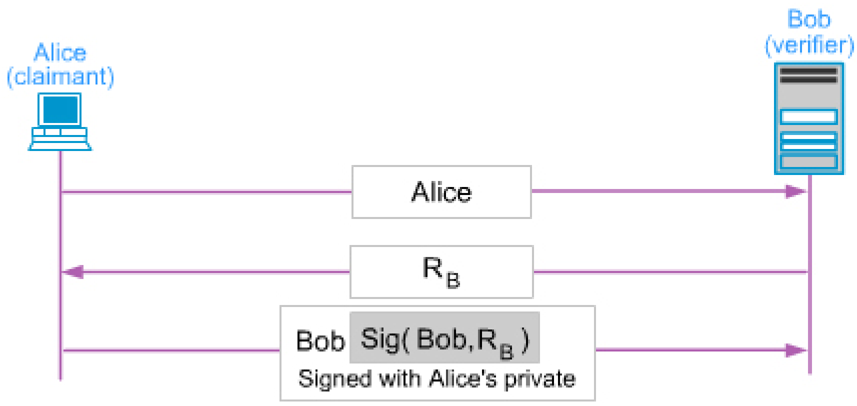
يؤدي ذلك إلى اكتشاف أن المدعي يعرف السر. لاحظ أنه لا يوجد أي تبادل للسر بين الطرفين المتصلين.



- **استخدام مشفر مزود بمفتاح غير متناظر:** يبين الإيضاح التالي فكرة هذه الطريقة، حيث أن: K_A هو المفتاح العام المرتبط بـ **Alice** و **R_B** هو التحدي:



- **استخدام التوقيع الالكتروني:** كما هو مبين في الشكل التالي:



الفصل العاشر
إدارة المفاتيح Key Management

الأهداف Objectives

يهدف هذا الفصل إلى التعريف بما يلي:

- ماهية مركز توزيع المفاتيح **Key-distribution center (KDC)** وكيفية عمله.
- شرح كيف يمكن لطرفين أن يستخدما بروتوكول معتمد على المفاتيح المتناظرة لخلق جلسة تبادل آمن بينهما دون الحاجة للمرور عبر خدمات المركز **KDC**.
- شرح كيفية عمل النظام **Kerberos**.
- شرح الحاجة لوجود سلطات لمنح التراخيص **Certification Authorities (CAs)** للمفاتيح العامة.
- ماهية البنية التحتية المعتمدة على المفاتيح العامة **Public-Key Infrastructure (PKI)** وشرح كيفية عملها.

1. توزيع المفاتيح المتناظرة Symmetric-Key Distribution

تعتبر طرق التشفير بمفتاح متناظر أكثر فعالية من طرق التشفير بمفتاح غير متناظر عند التعامل مع نصوص ورسائل كبيرة. لكن هذه الطرق تعاني من مشكلة توزيع المفتاح السري المشترك بين طرفي الاتصال.

كذلك الأمر، إذا رغب أحد الأشخاص بتبادل رسائل سرية مع **N** شخص آخر فهو يحتاج إلى **N** مفتاح مختلف. وإذا احتاجت مجموعة من **N** شخص للتخاطب فيما بينها، فإنها تحتاج إلى استخدام $N(N-1)$ مفتاح إذا كنا نرغب باستخدام مفتاح مختلف لكل جهة اتصال مختلفة، أو إلى استخدام $N(N/-1)\ 2$ إذا كنا نرغب باستخدام نفس المفتاح بين كل طرفين محددين وبالاتجاهين بينهما. يعني ذلك كله وجود عدد كبير من المفاتيح التي تحتاج إلى إدارة لعملية توزيعها بين الأشخاص المختلفين بشكل آمن لا يكشف عنها للأطراف الأخرى.

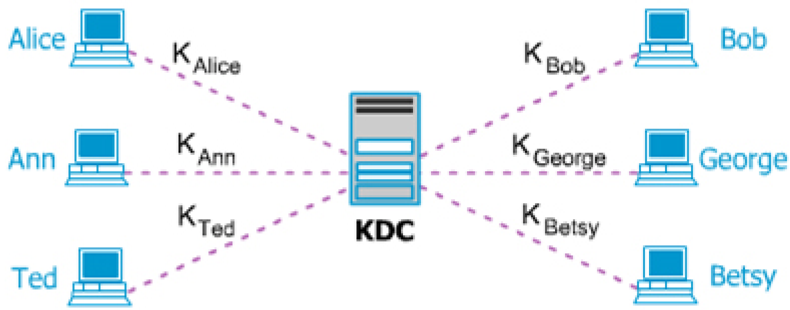
لتنفيذ عملية التوزيع دون أية اشكالات يمكن اللجوء إلى طرف ثالث موثوق **Trusted Third Party**. يجب أن تتوفر في هذا الطرف الخصائص التالية:

- يعرف جيداً كل الأشخاص المعنيين باستخدام المفاتيح.
- يتحقق من هوية الأشخاص ويدققها.
- يقوم بإدارة مفاتيح الأشخاص.

بالنسبة للمفاتيح المتناظرة، نسمي هذا الطرف الثالث بمركز توزيع المفاتيح **Key-Distribution Center (KDC)**. أما في حالة المفاتيح غير المتناظرة، فنسمي هذا الطرف بسلطة التراخيص **Certificate Authority**.

مركز توزيع المفاتيح Key-Distribution Center (KDC)

يهدف هذا المركز إلى التقليل من عدد المفاتيح التي نحتاج إليها للتراسل الآمن بين الأشخاص، كما هو مبين في الإيضاح التالي:



حيث يجري تعريف مفتاح سري بين كل شخص والمركز. ويجري التراسل وفقاً للآلية التالية:

1. يقوم الطرف الأول بإرسال طلب إلى المركز **KDC** لطلب فتح جلسة تراسل مؤقتة بينه وبين طرف ثاني.
2. يخبر المركز الطرف الثاني برغبة الطرف الأول.
3. إذا وافق الطرف الثاني على التراسل مع الطرف الأول، يتم إنشاء مفتاح جلسة **Session Key** أو بطاقة جلسة **Ticket Seaaion** بين الطرفين ويستخدم مفتاحهما السريين لإيصاله لهما.

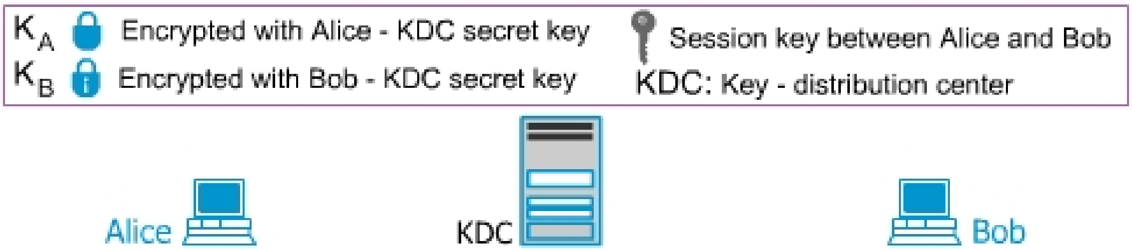
تختلف بطاقة الجلسة عن مفتاح الجلسة في أن البطاقة تتضمن إضافة إلى مفتاح الجلسة تعريفاً بالمرسل والمستقبل. ويجري التعامل بهما خلال فترة مؤقتة (أثناء جلسة الاتصال) ولمرة واحدة فقط.

عندما يتزايد عدد الأشخاص كثيراً، تصبح عملية إدارة المفاتيح صعبة ويتحول المركز **KDC** إلى مركز اختناق لمعالجة الطلبات الواردة إليه. لتجاوز هذه المشكلة، نحتاج إلى تواجد عدة مراكز **KDC**. ويمكن تقسيم هذه المراكز إلى عدة مجالات **Domains**.

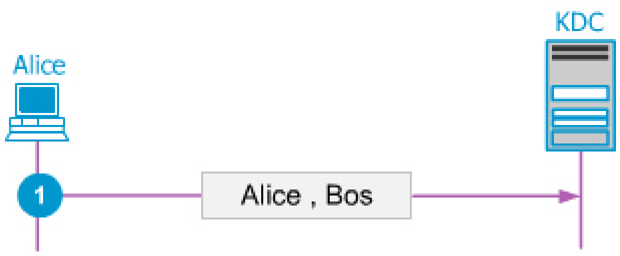
فعند الحاجة لإرسال رسالة إلى شخص من مجال مختلف، يجري الاتصال بمركز توزيع المفاتيح الخاص بالمرسل الذي يقوم بدوره بالاتصال مع مركز توزيع مفاتيح المجال الآخر لتوليد مفتاح جلسة بينهما. يمكن أيضاً ربط مراكز التوزيع فيما بينها هرمياً وفقاً لبنية نراها مناسبة.

بروتوكول KDC البسيط Simple KDC Protocol

إذا رغب شخصين (Alice and Bob) بالاتصال فيما بينها بشكل آمن، فكيف يمكن تحقيق ذلك؟ نبين فيما يلي كيف يمكن تعريف مفتاح جلسة K_{AB} بينهما للبدء بعدها بالاتصال الآمن المطلوب.

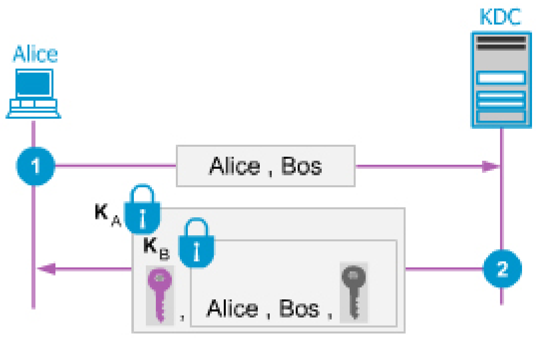


تتم عملية تعريف مفتاح الجلسة وفقاً للبرتوكول التالي:

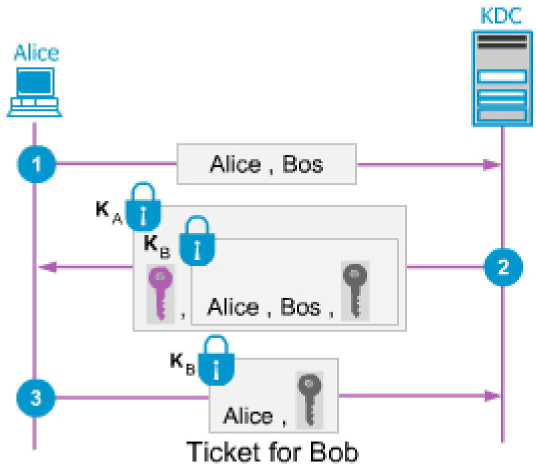


1. تقوم **Alice** بإرسال طلب إلى المركز **KDC** يتضمن هذا الطلب تعريفاً بهويتها وهوية الطرف الآخر الذي ترغب بالتواصل معه.
2. يقوم المركز بإنشاء مفتاح جلسة بين **Alice** و**Bob**.
3. ينشئ المركز بعدها بطاقة جلسة **Ticket** قابلة للقراءة من قبل **Bob** فقط لأنها مشفرة بمفتاحه. تتضمن هذه البطاقة مفتاح جلسة الاتصال السري مع **Alice**.
4. يرسل المركز البطاقة إلى **Alice** مشفرة بمفتاحها حتى لا يستطيع أحد آخر قراءتها. تقوم **Alice** بفكها للحصول على مفتاح الجلسة والبطاقة المشفرة بمفتاح **Bob**.

5. ترسل **Alice** البطاقة التي حصلت عليها إلى **Bob** والتي لن يستطيع أحد الاطلاع عليها لأنها مشفرة بمفتاحه. يقوم **Bob** بفك الرسالة للحصول على هوية **Alice** وعلى مفتاح الجلسة السري بينهما.

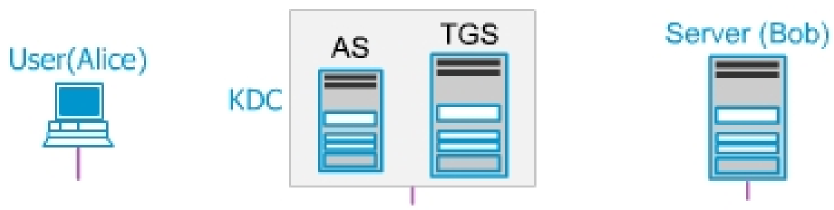


يتوفر لدى الطرفين الآن مفتاح الجلسة السري وبالتالي يمكنهما تبادل المعطيات بشكل آمن طيلة فترة جلسة الاتصال.



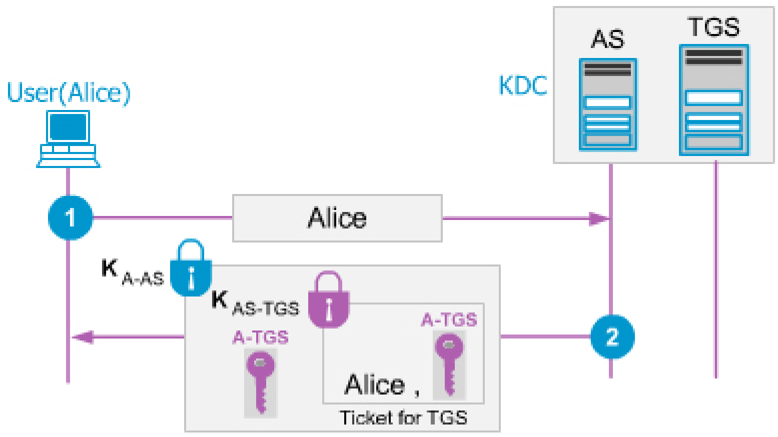
البرتوكول Kerberos

هذا البرتوكول هو برتوكول تحديد هوية **Protocol Authentication**. وهو مستخدم على نطاق واسع من قبل الكثير من أنظمة التشغيل. تم تصميمه من قبل معهد **MIT** بعد مروره بعدة إصدارات قبل أن نصل إلى صيغته الحالية. سوف نتحدث هنا عن النسخة الرابعة منه. يتضمن البرتوكول **Kerberos** التعامل مع ثلاث مخدمات، هي: مخدم تحديد الهوية **Authentication Server (AS)** ومخدم منح البطاقات **Ticket-granting Server (TGS)** والمخدم المتضمن للخدمات المطلوب الوصول إليها.



نبين فيما يلي كيفية عمل البرتوكول لتحديد الهوية وتسهيل الوصول إلى مخدم الخدمات:

1. يرسل الزبون **Alice** هويته إلى المخدم **AS**.
2. يقوم المخدم **AS** بإنشاء مفتاح مؤقت $K_{A\text{-}TGS}$ للاتصال مع المخدم **TGS** وبطاقة للتواصل الآمن مع المخدم **TGS** مشفرة بالمفتاح المؤقت. يرسل المخدم **AS** ذلك كله إلى **Alice** بعد تشفيره بمفتاح **Alice** المتناظر $K_{A\text{-}AS}$.

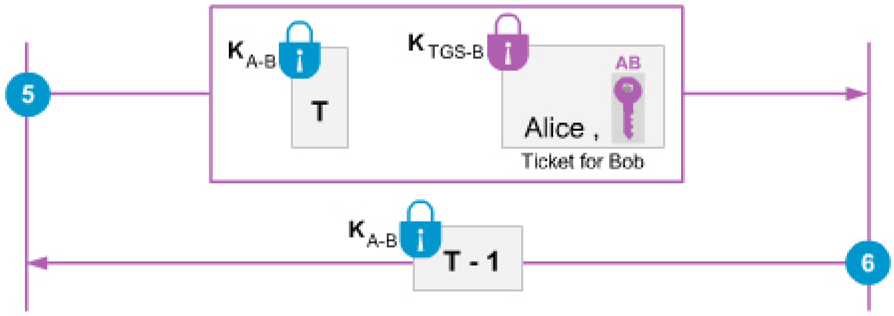


3. يقوم حاسب الزبون **Alice** من جهته بالطلب منه إدخال كلمة سر لإعادة إنشاء المفتاح المتناظر للزبون $K_{A\text{-}AS}$. ثم يجري استخدام هذا المفتاح لفك الرسالة والحصول بالتالي على: $K_{A\text{-}TGS}$ والبطاقة المرفقة التي ترسل بدورها إلى المخدم **TGS** مصحوبة باسم المخدم المطلوب الوصول إليه وهو **Bob** وطابع الزمن **Timestamp** بعد تشفير ذلك كله بالمفتاح $K_{A\text{-}TGS}$.

4. يرسل المخدم **TGS** إلى الزبون: بطاقة الزبون التي تتضمن مفتاح الجلسة $K_{A\text{-}B}$ مشفرة بالمفتاح $K_{A\text{-}TGS}$ وبطاقة المخدم التي تتضمن مفتاح الجلسة $K_{A\text{-}B}$ وهوية الزبون **Alice** ويجري تشفير هذه البطاقة الثانية بالمفتاح $K_{B\text{-}TGS}$ المؤقت.

5. يرسل الزبون إلى المخدم Bob البطاقة المشفرة بالمفتاح K_{B-TGS} والطابع الزمن Timestamp مشفراً بالمفتاح K_{A-B}.

6. يقوم المخدم Bob بفك الرسالة بالمفتاح K_{A-B} ويستخدم مفتاح الجلسة للحصول على الطابع الزمني ثم يرسل رسالة مصافحة متضمنة قيمة الطابع الزمني-1 إلى الزبون.



2. توزيع المفاتيح العامة Public-Key Distribution

في عالم التشفير باستخدام مفاتيح غير متناظرة، لا نحتاج إلى استخدام مفتاح مشترك متناظر. إذا أراد طرف أول إرسال معلومات إلى طرف ثاني فإنه يحتاج إلى معرفة المفتاح العام له فقط، وهذا المفتاح متاح للجميع. وبالعكس، إذا أراد الطرف الثاني الرد على الطرف الأول فإنه يحتاج إلى معرفة المفتاح العام له الذي هو متاح للجميع أيضاً. مع المشفرات المعتمدة على اسخدام مفاتيح غير متناظرة، يقوم كل طرف بالاحتفاظ بمفتاحه الخاص ويعلن للجميع عن مفتاحه العام.

على اعتبار أنه يجب توزيع المفاتيح العامة، سنتحدث فيما يلي عن طريقة توزيع هذه المفاتيح:

1. التصريح العلني **Public Announcement**.
2. المركز الموثوق **Trusted Center**.
3. المركز الموثوق المضبوط **Controlled Trusted Center**.
4. سلطة الترخيص **Certification Authority**.
5. المعيار **X.509**.
6. البنية التحتية للمفاتيح العامة **Public-Key Infrastructures (PKI)**.

التصريح العلني Public Announcement

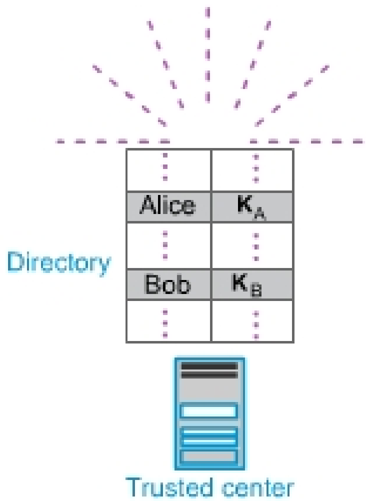
يمكن أن يتبادر إلى الذهن أنه يكفي أن نصرح للعموم عن المفتاح العام، فيمكن مثلاً لشخص أن يضع مفتاحه العام على موقه الالكتروني أو الصحف أو غير ذلك من الطرق. تعتبر هذه الطريقة طريقة غير آمنة وتعطي المجال للآخرين للتزوير.

على سبيل المثال:

يمكن لشخص أن يقوم بنفس التصريح العام، وقبل أن يقوم صاحب هذا التصريح الحقيقي بأي عمل مضاد، فإن هذا الطرف المدعي يمكن أن يبدأ بتلقي معلومات موجهة إليه على أنه الشخص المفترض فيه تلقيها. يمكن أيضاً لطرف خارجي أن يوقع رسالة بمفتاحه الخاص ويقنع الجميع بأنه الشخص الثاني المفترض أن يقوم بالتوقيع الرسالة وليس هذا الشخص الدخيل. وغير ذلك من الإشكالات الممكن حدوثها.

المركز الموثوق Trusted Center

الوسيلة الأكثر أمناً لتبادل المفاتيح العامة، هي التعامل مع مركز موثوق. يمتلك هذا المركز سجل للمفاتيح العامة. يشبه هذا السجل سجل أرقام الهاتف وهو عرضة للتعديل الديناميكي. يمكن لأي شخص أن يختار مفتاح عام ومفتاح خاص، ثم يحتفظ بالمفتاح الخاص معه ويرسل المفتاح العام إلى المركز الموثوق ليصار إلى إدارجه في السجل وذلك بعد التحقق من هويته والتأكد من شخصيته. يمكن الإعلان عن هذا السجل من قبل المركز الذي يمكنه أيضاً التجاوب مع أي طلب له علاقة بمفتاح عام. يبين الشكل التالي طريقة العمل هذه:



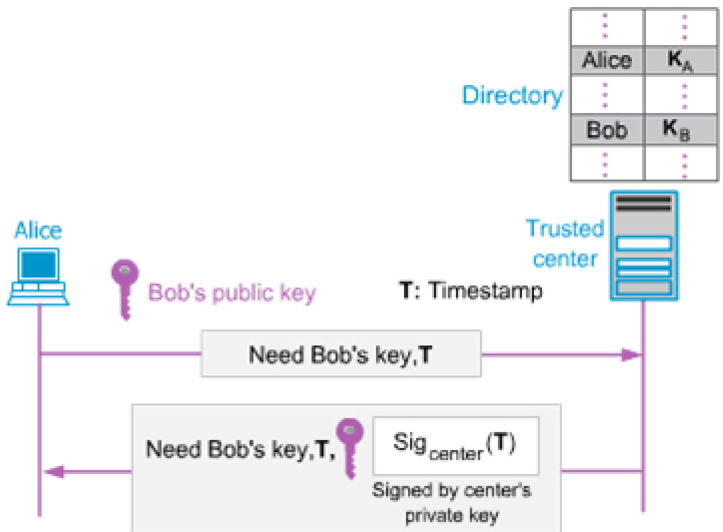
المركز الموثوق المضبوط Controlled Trusted Center

يعمل بنفس طريقة عمل المركز الموثوق مع إضافة تدابير أمنية على مسألة توزيع المفاتيح العامة. يمكن أن تترافق عملية التصريح عن المفاتيح السرية مع عملية إدارج للطابع الزمني وأن يتم توقيعها من قبل سلطة مشرفة لمنع التقاطها وتعديلها.

إذا أرادت **Alice** مثلاً أن تعرف المفتاح العام لـ **Bob** فيجب عليها أن تطلبه من المركز الموثوق الذي يتبع إليه **Bob**. يجب أن يتضمن الطلب اسم **Bob** والطابع الزمني.

عند استجابة المركز الموثوق فإنه يقوم بإرسال المفتاح العام لـ **Bob** والطلب الأصلي وطابع زمني موقع باستخدام المفتاح الخاص للمركز.

تقوم بعدها **Alice** باستخدام المفتاح العام للمركز المعروف للجميع للتحقق من صحة الطابع الزمني. فإذا كان الطابع الزمني صحيحاً فيمكن استخلاص المفتاح العام لـ **Bob**.



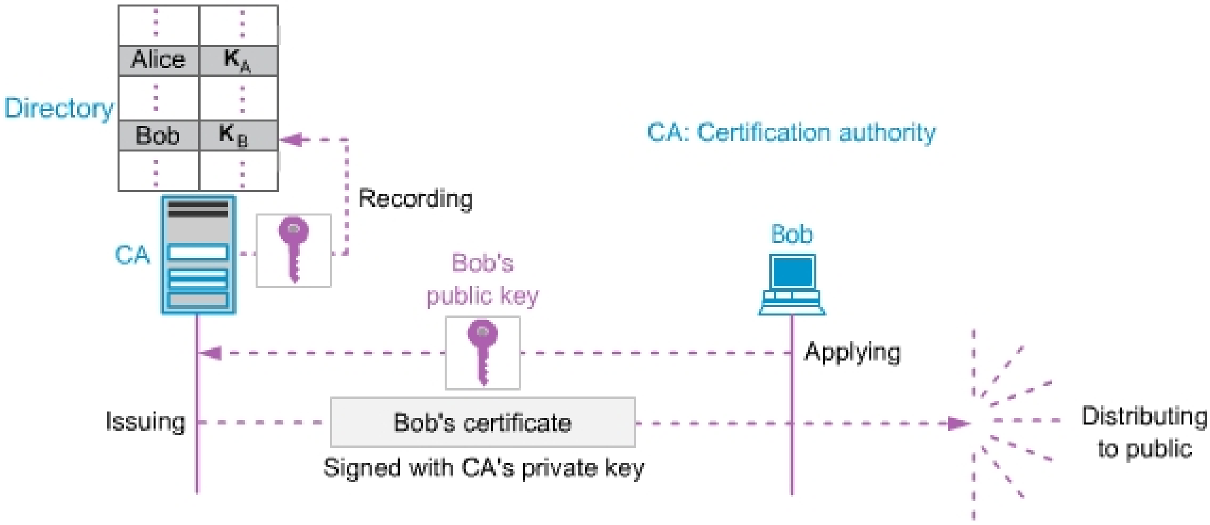
سلطة الترخيص Certification Authority

يعاني الحل السابق من مشكلة أساسية متمثلة في القدرة على معالجة العدد الكبير من الطلبات الممكن ورودها. الحل البديل والسليم للأمر هو في بناء شهادات مفاتيح عامة **Public-key certificates**. أي شخص (مثل **Bob**) يريد أمرين اثنين، هما:

1. أن يعرف الآخرين بمفتاحه العام.
2. وأن لا يستطيع أي شخص آخر أن يقبل مفتاح عام آخر مزور على أنه مفتاحه (أي مفتاح **Bob**).

يمكن لـ **Bob** أن يذهب إلى سلطة ترخيص **Certification Authority (CA)** وهي مؤسسة أو منظمة لبناء المفاتيح العامة وإصدارها ضمن شهادة **Certificate**. تتمتع هذه السلطة بمفتاح عام معروف جيداً ولا يمكن تزويره. تقوم هذه السلطة بالتحقق من هوية **Bob**، ثم تطلب المفتاح العام لـ **Bob** وتكتبه ضمن الشهادة الصادرة. لمنع تزوير الشهادات نفسها، تقوم السلطة بتشفيرها بواسطة المفتاح الخاص بها. يمكن لـ **Bob** بعدها أن يشحن الشهادة الموقعة والممنوحة له.

ويمكن أيضاً لكل من يريد معرفة المفتاح العام لـ **Bob** أن يجلب شهادته الموقعة ويستخدم المفتاح العام للسلطة المانحة لاستخراج المفتاح العام المرتبط بـ **Bob**. يعبر الشكل التالي عن هذا المفهوم.

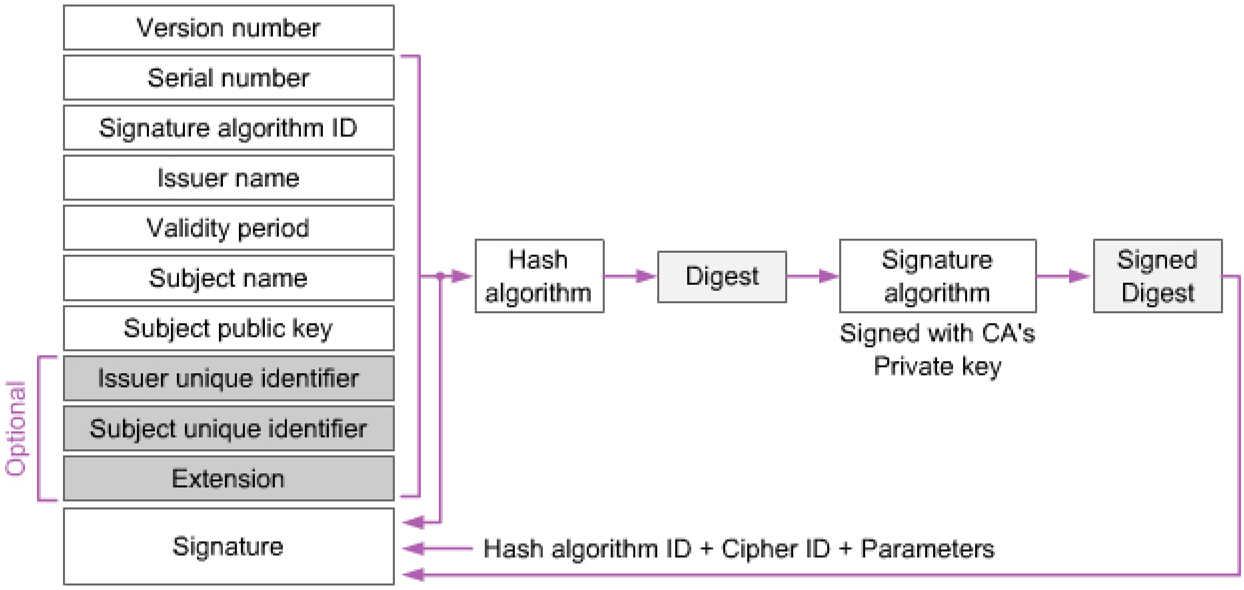


المعيار X.509

على الرغم من أن سلطة الترخيص ساعدت على حل مشكلة تزوير المفاتيح العامة، إلا أنها خلقت مشكلة جديدة لها علاقة بصيغة الشهادات التي يجب أن تصدرها. يجب أن تكون هذه الصيغة موحدة حتى تستطيع الأدوات البرمجية إيجاد المفتاح العام ضمنها واستخدامه.

لحل هذه المشكلة الجديدة، قام المعهد **ITU** بتصميم المعيار **X.509** للصيغة العامة للشهادة وبنيتها.

يبين الشكل التالي شكل الشهادة وصيغتها العامة:



يوجد ضمن الشهادة عدة حقول، نذكر منها:

- الحقل "**رقم الاصدار**" **Version number** لتحديد رقم نسخة المعيار **X.509** للشهادة. يبدأ هذا الرقم من **0** ويصل إلى الرقم **2** (وهو النسخة الثالثة والحالية من المعيار).
- الحقل "**الرقم التسلسلي**" **Serial number** لتحديد الرقم التسلسلي المسند للشهادة. يجب أن يكون هذا الرقم وحيد لكل شهادة مصدرة.
- الحقل "**معرف خوارزمية التوقيع**" **Signature algorithm ID** لتحديد خوارزمية توقيع الشهادة. كما يجري تعريف أي وسيط يحتاجه هذه الجوارزمية.
- الحقل "**اسم المصدر**" **Issuer name** لتحديد اسم لاسلطة مصدرة الشهادة.
- الحقل "**مدة الصلاحية**" **Validity Period** لتحديد الفترة الرزمنية التي تكون ضمنها الشهادة صالحة.
- الحقل "**اسم المالك**" **Subject Name** لتحديد اسم مالك الشهادة على شكل سلسلة حروف.
- الحقل "**المفتاح العام للمالك**" **Subject public key** لتحديد المفتاح العام للمالك. ويشكل هذا المفتاح قلب الشهادة. كما يحدد ضمن هذا الحقل اسم خوارزمية المفتاح العام المرتبطة به (على سبيل المثال: **RSA**) ووسطائها.
- الحقل الاختياري "**معرف المصدر**" **Issuer unique identifier** للتمييز بين مصدرين للشهادات لهما نفس الاسم.
- الحقل الاختياري "**معرف المالك**" **Subject unique identifier** للتمييز بين عدة مالكين

يحملون نفس الاسم.

- الحقل الاختياري "**إضافات**" **Extensions** للسماح للمالك بإضافة معلومات خاصة به.
- الحقل "**التوقيع**" **Signature** وهو مؤلف من ثلاثة مقاطع. يتضمن المقطع الأول بقية حقول الشهادة. أما المقطع الثاني فيتضمن بصمة المقطع الأول مشفرة باستخدام المفتاح العام للسلطة المصدرة. ونجد في المقطع الثالث معرف خوارزمية التشفير المستعملة لبناء المقطع الثاني.

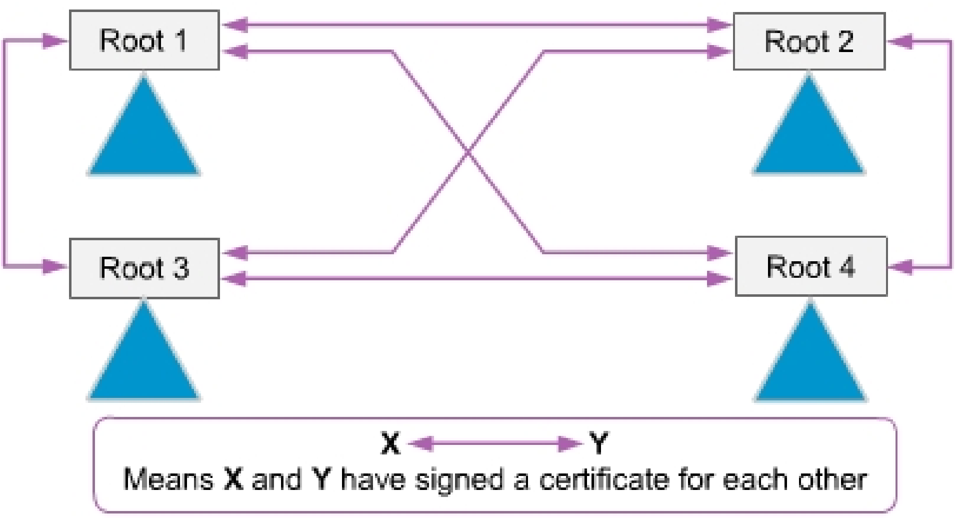
يمكن تجديد الشهادة الممنوحة قبل نهاية مدة صلاحيتها في حال عدم وجود أية إشكالات. تشبه هذه العملية مدة تجديد صلاحية بطاقات الدفع الالكتروني.

كما تستخدم الشهادة لتنفيذ عمليات التحقق على الشكل التالي: نقوم باستخدام خوارزمية التهشير المشار إليها ضمن الشهادة ونطبقها على جميع حقول الشهادة لحساب البصمة ومقارنتها مع البصمة المخزنة ضمن الشهادة. إذا تطابقت القيمتين فالشهادة صالحة ولم يجر التلاعب بها.

يمكن أيضاً سحب الشهادة وإلغائها قبل إنقضاء فترة الصلاحية لأسباب عديدة ويترتب نتيجة ذلك عدة أمور نذكر منها: إيقاف عمل المفتاح الخاص لمالك الشهادة، وعدم التعامل معه مستقبلاً. تقوم سلطة الترخيص بإصدار قائمة بالشهادات الملغاة وبالتالي يجب على المستخدم أن يتأكد من أن الشهادة التي يتعامل معها ملغاة أم لا قبل الوثوق بها واستخدامها.

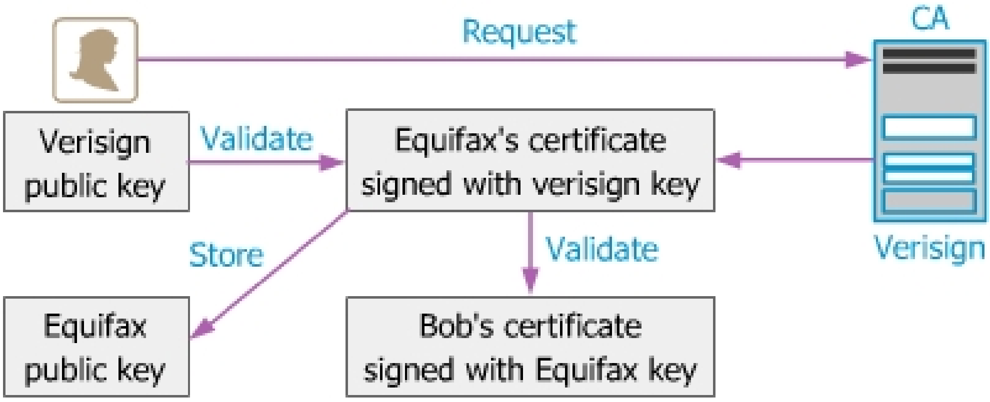
البنية التحتية للمفاتيح العامة Public-Key Infrastructures (PKI)

تعتبر هذه البنية التحتية نموذجاً لإنشاء وتوزيع وإلغاء الشهادات. ويمكن أن يتعامل المستخدمين مع سلطات ترخيص مختلفة. يمكن الربط بين هذه السلطات المختلفة بأن تقوم كل واحدة منها بتخزين شهادات لبقية السلطات الأخرى كما هو مبين في الشكل:



كما يمكن الربط بين سلطات الترخيص بشكل هرمي بدلاً من البنية المصفوفية المبينة في الشكل السابق.

يوضح الشكل التالي سيناريو العمل لطلب شهادة شخص مسجل لدى سلطة مختلفة عن سلطة شخص يرغب بالتعامل معه:



يوضح هذا السيناريو كيف يجري النقاش بين السلطتين للحصول على شهادة **Bob** المخزنة لدى السلطة **Equifax** والتي تحتاجها **Alice** التي تملك شهادة مصدرة من قبل السلطة **Verisign**.

الجزء الرابع

تحقيق بيئة آمنة

Developping Secure Environment

الفصل الحادي عشر
سياسات أمن المعلومات
Information Security Policies

1. مقدمة Introduction

يمتد تأثير تقانات الاتصالات والمعلومات امتداداً واسعاً داخل جميع النشاطات الاقتصادية والاجتماعية، إذ ترتبط تنمية تقانات الاتصالات والمعلومات ارتباطاً عضوياً بمختلف مناحي التنمية في جميع القطاعات، من الإنتاج الصناعي إلى الخدمات.

وتؤدي هذه التقانات دوراً أساسياً في تطوير المجتمعات الحديثة ("مجتمع المعرفة")، فمفهوم "المعلومات" يرتبط بجميع النشاطات المؤدية إلى نشر وإنتاج المعرفة، من التعليم ووسائط الإعلام المكتوبة والمسموعة والمرئية، وصولاً إلى الإنتاج الثقافي والفكر، ويعد قطاع الاتصالات والمعلومات من أسرع القطاعاتتطوراً في العالم، وقد بات تطوير هذا القطاع ضرورة للإسهام في التنمية الاقتصادية والاجتماعية للبلدان المختلفة.

نعرض في هذا الفصل السياسات العامة التي يجب توفرها في أي بلد لتحصين بيئة العمل فيه وجعلها آمنة.

2. سياسات عامة General Policies

تلعب بعض الجهات المختصة دور المحفز من أجل تطوير الاتصالات والمعلومات وتطوير المحتوى الرقمي وبناء القدرات، بالإضافة إلى خلق وإيجاد البيئة التمكينية المساعدة على الإنتاج والإبداع في هذا المجال، ولا بد لذلك من التشارك مع كافة أصحاب المصلحة من القطاع العام والخاص ومؤسسات المجتمع الأهلي، وتؤدي الحكومات الدور التنسيقي على نحو يضمن المصالح العامة، ولكن تبقى هي اللاعب الرئيسي والمحفز، من خلال وضع رؤية تنموية تلتزم بتسخير إمكانيات هذه التقانات للنهوض بأهداف التنمية الاقتصادية والاجتماعية في سوريا، على نحو يؤدي إلى:

- تقديم خدمات الاتصالات والمعلومات، من الهاتف الثابت والنقّال، وتبادل المعطيات، والنفاذ إلى الإنترنت، محلياً ودولياً، بجودة عالية وأجور مقبولة للأفراد والمؤسسات أينما وجدوا، وعلى نحو يزيد من فعاليتهم الاقتصادية والاجتماعية.
- توفير منظومات الحواسيب ومعالجة المعلومات بأشكالها المختلفة لجميع المؤسسات والفعاليات، على نحو يمكّنها من معالجة المعلومات وفق احتياجاتها، بغية رفع مردوديتها وقدرتها التنافسية.
- بناء قطاع اقتصادي وصناعي يتمحور حول تقانات الاتصالات والمعلومات، وبوجه

خاص صناعة البرمجيات وتطوير المضمون على الإنترنت.

- استخدام نظم الاتصالات والمعلومات لدعم التنمية الاجتماعية، وخاصة لزيادة مشاركة المواطن في الحياة العامة، وإضفاء الشفافية على الإجراءات الحكومية، وتبسيط هذه الإجراءات.
- نشر وإنتاج المعرفة باستخدام تقانات الاتصالات والمعلومات.
- توفير الإطار التشريعي اللازم لتطوير واستخدام تقانات الاتصالات والمعلومات، وتنظيم قطاع الاتصالات والمعلوماتية.

ويمكن تقسيم العمل المطلوب إلى عدة محاور والتي نذكر من أهمها محور البيئة القانونية والتشريعية. ويمكن إنجازها من خلال محاور العمل التالية:

- إصدار قانون التوقيع الالكتروني وهيئة خدمات الشبكة، متضمناً منح تراخيص لجهات تقديم خدمات التوقيع الالكتروني، ومنح وإدارة أسماء النطاقات العليا للانترنت.
- إصدار قانون التعاملات الالكترونية، متضمنا الأسس العامة الناظمة للمعاملات الالكترونية والعمليات التجارية الجارية بالوسائل الالكترونية، وذلك بالتعاون مع وزارة الاقتصاد والتجارة.
- إصدار قانون حماية حقوق المؤلف المتضمن حماية المصنفات الالكترونية، بالتعاون مع وزارة الثقافة.
- إصدار قانون الصحافة الالكترونية وتنظيم عمل النشر الالكتروني وأصول منح التوصيف الصحفي، متضمناً شروط النشر الالكتروني، بالتعاون مع وزارة الإعلام.
- قانون حماية السرية والخصوصية، والذي يضمن حقوق سرية وخصوصية المعلومات الشخصية، ويضبط عملية إفشاء ونشر هذه المعلومات.
- قانون الجريمة الحاسوبية، الذي يحدد أنواع الجرائم المرتكبة على الشبكات الحاسوبية من اختراق للأنظمة والمواقع الالكترونية وعمليات القرصنة، بالإضافة إلى نشر المعلومات المؤدية إلى الجريمة، والعقوبات الخاصة بكل نوع من الجرائم السابقة.
- بناء مجتمع المعلومات.

3. خاتمة Conclusion

تحدثنا في هذا الفصل عن البنية القانونية والتشريعية الواجب توفرها في المجتمع قبل الحديث عن ضرورة توفير وسائل تشفير للمعلومات.

نعتقد أن توفر مثل هذه البيئة، يشكل أساساً لا بد منه من أجل تطوير قطاعي التقانة والاتصالات في أي بلد.

References المراجع العلمية

- Introduction to Cryptography and Network Security, Behrouz A. Forouzan, McGraw-Hill International Edition.
- Computer Security and Cryptography, Alan G. Konheim, Wiley, 2007.
- Applied Cryptography, Bruce Schneier, John Wiley & Sons, Inc., Second edition, 1996.
- Cryptography and Network Security Principles and Practices, William Stallings, Prentice Hall, Fourth edition, 2005.
- Cryptography: Theory and Practice, Douglas Stinson, CRC Press LLC, 1995.
- User's Guide to Cryptography and Standards, Alexander W. DENT and Chris J. Mitchell, Artech House – Computer Security Series, 2005.