Security II: Cryptography

Markus Kuhn

Computer Laboratory, University of Cambridge

http://www.cl.cam.ac.uk/teaching/1314/SecurityII/

These notes are provided as an aid for following the lectures, and are not a substitute for attending

Lent 2014 – Part II

Related textbooks

Main reference:

- Jonathan Katz, Yehuda Lindell:
  Introduction to Modern Cryptography
  Chapman & Hall/CRC, 2008 (new edition announced for March 2014)

Further reading:

- Christof Paar, Jan Pelzl:
  Understanding Cryptography
  Springer, 2010
  http://www.springerlink.com/content/978-3-642-04100-6/
  http://www.crypto-textbook.com/
- Douglas Stinson:
  Cryptography – Theory and Practice
  3rd ed., CRC Press, 2005
- Menezes, van Oorschot, Vanstone:
  Handbook of Applied Cryptography
  CRC Press, 1996
  http://www.cacr.math.uwaterloo.ca/hac/

Encryption schemes

Encryption schemes are algorithm triples $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

Private-key (symmetric) encryption scheme

- $K \leftarrow \mathsf{Gen}$ key generation
- $C \leftarrow \mathsf{Enc}_K(M)$ encryption
- $M = \mathsf{Dec}_K(C)$ decryption

Public-key (asymmetric) encryption scheme

- $(PK, SK) \leftarrow \mathsf{Gen}$
- $C \leftarrow \mathsf{Enc}_{PK}(M)$
- $M = \mathsf{Dec}_{SK}(C)$

Probabilistic algorithms: Gen and (often also) Enc access a random-bit generator that can toss coins (uniformly distributed, independent).

Notation: $\leftarrow$ assigns the output of a probabilistic algorithm, $:=$ that of a deterministic algorithm.

When is an encryption scheme "secure"?

If no adversary can ...

- ... find out the key K?
- ... find the plaintext message M?
- ... determine any character/bit of M?
- ... determine any information about M from C?
- ... compute any function of the plaintext M from ciphertext C?
  $\Rightarrow$ "semantic security"

Note about message length: we explicitly do *not* worry here about the adversary being able to infer something about the length m of the plaintext message M by looking at the length n of the ciphertext C.

Therefore, we consider for the following security definitions only messages of *fixed* length m.

Variable-length messages can always be extended to a fixed length, by padding, but this can be expensive. It will depend on the specific application whether the benefits of fixed-length padding outweigh the added transmission cost.

What capabilities may the adversary have?

- unlimited / polynomial / realistic ($\ll 2^{80}$ steps) computation time?
- only access to ciphertext C?
- access to some plaintext/ciphertext pairs (M, C) with $C \leftarrow \mathrm{Enc}_K(M)$?
- how many applications of K can be observed?
- ability to trick the user of Enc_K into encrypting some plaintext of the adversary's choice and return the result?
  (“oracle access” to Enc)
- ability to trick the user of Dec_K into decrypting some ciphertext of the adversary's choice and return the result?
  (“oracle access” to Dec)?
- ability to modify or replace C en route?
  (not limited to eavesdropping)

Wanted: Clear definitions of what security of an encryption scheme means, to guide both designers and users of schemes, and allow proofs.

1. **Symmetric encryption**
2. **Message authenticity**
3. **Authenticated encryption**
4. **Asymmetric encryption**
5. **Number theory**
6. **RSA trapdoor function**

Recall: perfect secrecy

Perfect secrecy

An encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{M}$ is *perfectly secret* if for every probability distribution over $\mathcal{M}$, every message $M \in \mathcal{M}$, and every ciphertext $C \in \mathcal{C}$ with $P(C) > 0$ we have

$$P(M|C) = P(M).$$

In other words: even an eavesdropper with unlimited computational power cannot learn anything about M by looking at C that they didn't already know in advance about M $\Rightarrow$ eavesdropping C has no benefit.

Recall: one-time pad

Shannon's theorem:

Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme over a message space $\mathcal{M}$ with $|\mathcal{M}| = |\mathcal{K}| = |\mathcal{C}|$. It is perfectly secret if and only if

1. Gen chooses every K with equal probability $1/|\mathcal{K}|$;
2. for every $M \in \mathcal{M}$ and every $C \in \mathcal{C}$, there exists a unique key $K \in \mathcal{K}$ such that $C := \mathsf{Enc}_K M$.

The standard example of a perfectly-secure symmetric encryption scheme:

One-time pad

$$\begin{array}{rll} \mathsf{Gen}: & K \in_{\mathsf{R}} \{0,1\}^m & (m \text{ uniform, independent coin tosses}) \\ \mathsf{Enc}: & C := K \oplus M & (\text{bit-wise XOR}) \\ \mathsf{Dec}: & M := K \oplus C & \end{array}$$

Private-key (symmetric) encryption

A **private-key encryption scheme** is a tuple of probabilistic polynomial-time algorithms (Gen, Enc, Dec) and sets $\mathcal{K}, \mathcal{M}, \mathcal{C}$ such that

- the **key generation algorithm** Gen receives a security parameter ℓ and outputs a key $K \leftarrow \mathsf{Gen}(1^\ell)$, with $K \in \mathcal{K}$, key length $|K| \geq \ell$;
- the **encryption algorithm** Enc maps a key K and a plaintext message $M \in \mathcal{M} = \{0,1\}^m$ to a ciphertext message $C \leftarrow \mathsf{Enc}_K(M)$;
- the **decryption algorithm** Dec maps a key K and a ciphertext $C \in \mathcal{C} = \{0,1\}^n$ $(n \geq m)$ to a plaintext message $M := \mathsf{Dec}_K(C)$;
- for all ℓ, $K \leftarrow \mathsf{Gen}(1^\ell)$, and $M \in \{0,1\}^m$: $\mathsf{Dec}_K(\mathsf{Enc}_K(M)) = M$.

Notes:

A "polynomial-time algorithm" has constants a, b, c such that the runtime is always less than $a \cdot \ell^b + c$ if the input is ℓ bits long. (think Turing machine)

Technicality: we supply the security parameter ℓ to Gen here in unary encoding (as a sequence of ℓ "1" bits: 1^ℓ), merely to remain compatible with the notion of "input size" from computational complexity theory. In practice, Gen usually simply picks ℓ random bits $K \in_{\mathsf{R}} \{0,1\}^\ell$.

Security definitions for encryption schemes

We define security via the rules of a game played between two players:

- a challenger, who uses an encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$
- an adversary $\mathcal{A}$, who tries to demonstrate a weakness in Π.

Most of these games follow a simple pattern:

1. the challenger uniformly picks at random a secret bit $b \in_{\mathsf{R}} \{0,1\}$
2. $\mathcal{A}$ interacts with the challenger according to the rules of the game
3. At the end, $\mathcal{A}$ has to output a bit b'.

The outcome of such a game $X_{\mathcal{A},\Pi}(\ell)$ is either

- $b = b' \Rightarrow \mathcal{A}$ won the game, we write $X_{\mathcal{A},\Pi}(\ell) = 1$
- $b \neq b' \Rightarrow \mathcal{A}$ lost the game, we write $X_{\mathcal{A},\Pi}(\ell) = 0$

Advantage

One way to quantify $\mathcal{A}$'s ability to guess b is

$$\mathsf{Adv}_{X_{\mathcal{A},\Pi}(\ell)} = \left| P(b = 1 \text{ and } b' = 1) - P(b = 0 \text{ and } b' = 1) \right|$$

Negligible advantage

Security definition

An encryption scheme Π is considered "X secure" if for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ there exists a "negligible" function negl such that

$$P(X_{\mathcal{A},\Pi}(\ell) = 1) < \frac{1}{2} + \mathrm{negl}(\ell).$$

Some authors prefer the equivalent definition with

$$\mathrm{Adv}_{X_{\mathcal{A},\Pi}(\ell)} < \mathrm{negl}(\ell).$$

Negligible functions

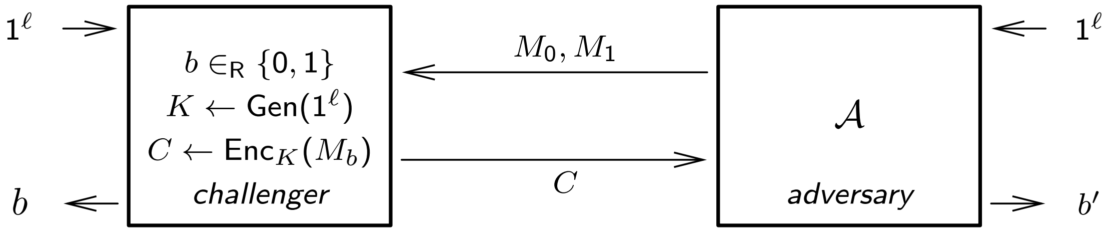
A function negl(ℓ) is "negligible" if it converges faster to zero than any polynomial over ℓ does, as $\ell \to \infty$.

In practice: We want negl(ℓ) to drop below a small number (e.g., 2^{-80} or 2^{-100}) for modest key lengths ℓ (e.g., $\log_{10} \ell \approx 2 \ldots 3$). Then no realistic opponent will have the computational power to repeat the game often enough to win at least once more than what is expected from random guessing.

Indistinguishability in the presence of an eavesdropper

Private-key encryption scheme $\Pi = (\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, $\mathcal{M} = \{0,1\}^m$, security parameter ℓ.

Experiment/game $\mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(\ell)$:



1^ℓ

$b \in_R \{0,1\}$

$K \leftarrow \mathrm{Gen}(1^\ell)$

$C \leftarrow \mathrm{Enc}_K(M_b)$

challenger

b

M_0, M_1

C

$\mathcal{A}$

adversary

1^ℓ

b'

Setup:

1. The challenger generates a bit $b \in_R \{0,1\}$ and a key $K \leftarrow \mathrm{Gen}(1^\ell)$.
2. The adversary $\mathcal{A}$ is given input 1^ℓ

Rules for the interaction:

1. The adversary $\mathcal{A}$ outputs a pair of messages: $M_0, M_1 \in \{0,1\}^m$.
2. The challenger computes $C \leftarrow \mathrm{Enc}_K(M_b)$ and returns C to $\mathcal{A}$

Finally, $\mathcal{A}$ outputs b'. If $b' = b$ then $\mathcal{A}$ has succeeded $\Rightarrow \mathrm{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(\ell) = 1$

Indistinguishability in the presence of an eavesdropper

Definition: A private-key encryption scheme Π has *indistinguishable encryption in the presence of an eavesdropper* if for all probabilistic, polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl, such that

$$P(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(\ell) = 1) \leq \frac{1}{2} + \mathsf{negl}(\ell)$$

In other words: as we increase the security parameter ℓ, we quickly reach the point where no eavesdropper can do significantly better than just randomly guessing b.

Pseudo-random generator I

$G : \{0,1\}^n \rightarrow \{0,1\}^{e(n)}$ where $e(\cdot)$ is a polynomial (expansion factor)

Definition

G is a **pseudo-random generator** if both

1. $e(n) > n$ for all n (expansion)
2. for all probabilistic, polynomial-time distinguishers D there exists a negligible function negl such that

$$|P(D(r) = 1) - P(D(G(s)) = 1)| \leq \mathsf{negl}(n)$$

where both $r \in_{\mathsf{R}} \{0,1\}^{e(n)}$ and the seed $s \in_{\mathsf{R}} \{0,1\}^n$ are chosen at random, and the probabilities are taken over all coin tosses used by D and for picking r and s.

Pseudo-random generator II

A brute-force distinguisher D would enumerate all 2^n possible outputs of G, and return 1 if the input is one of them.

It would achieve

$$P(D(G(s)) = 1) = 1$$
$$P(D(r) = 1) = \frac{2^n}{2^{e(n)}}$$

the difference of which converges to 1, which is not negligible.

But a brute-force distinguisher has a exponential run-time $O(2^n)$, and is therefore excluded!

We do not know how to prove that a given algorithm is a pseudo-random generator, but there are many algorithms that are widely believed to be.

Some constructions are pseudo-random generators if another well-studied problem is not solvable in polynomial time.

Encrypting using a pseudo-random generator

We define the following fixed-length private-key encryption scheme:

$\Pi_{\mathsf{PRG}} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

Let G be a **pseudo-random generator** with expansion factor $e(\cdot)$, $\mathcal{K} = \{0,1\}^\ell$, $\mathcal{M} = \mathcal{C} = \{0,1\}^{e(\ell)}$

- Gen: on input 1^ℓ chose $K \in_{\mathsf{R}} \{0,1\}^\ell$ randomly
- Enc: $C := G(K) \oplus M$
- Dec: $M := G(K) \oplus C$

Such constructions are known as "stream ciphers".

We can prove that Π_{PRG} has "indistinguishable encryption in the presence of an eavesdropper" assuming that G is a pseudo-random generator: if we had a polynomial-time adversary $\mathcal{A}$ that can succeed with non-negligible advantage against Π_{PRG}, we can turn that using a polynomial-time algorithm into a polynomial-time distinguisher for G, which would violate the assumption.

Security proof for a stream cipher

Claim: Π_{PRG} has indistinguishability in the presence of an eavesdropper if G is a pseudo-random generator.

Proof: (outline) If Π_{PRG} did not have indistinguishability in the presence of an eavesdropper, there would be an adversary $\mathcal{A}$ for which

$$\epsilon(\ell) := P(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi_{\mathsf{PRG}}}(\ell) = 1) - \frac{1}{2}$$

is not negligible.

Use that $\mathcal{A}$ to construct a distinguisher D for G:

- receive input $W \in \{0,1\}^{e(\ell)}$
- pick $b \in_{\mathsf{R}} \{0,1\}$
- run $\mathcal{A}(1^\ell)$ and receive from it $M_0, M_1 \in \{0,1\}^{e(\ell)}$
- return $C := W \oplus M_b$ to $\mathcal{A}$
- receive b' from $\mathcal{A}$
- return 1 if $b' = b$, otherwise return 0

Now, what is $|P(D(r) = 1) - P(D(G(K)) = 1)|$?

Security proof for a stream cipher (cont'd)

What is $|P(D(r) = 1) - P(D(G(K)) = 1)|$?

- What is $P(D(r) = 1)$?
  Let $\tilde{\Pi}$ be an instance of the one-time pad, with key and message length $e(\ell)$, i.e. compatible to Π_{PRG}. In the $D(r)$ case, where we feed it a random string $r \in_{\mathsf{R}} \{0,1\}^{e(n)}$, then from the point of view of $\mathcal{A}$ being called as a subroutine of $D(r)$, it is confronted with a one-time pad $\tilde{\Pi}$. The perfect secrecy of $\tilde{\Pi}$ implies $P(D(r) = 1) = \frac{1}{2}$.
- What is $P(D(G(K)) = 1)$?
  In this case, $\mathcal{A}$ participates in the game $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi_{\mathsf{PRG}}}(\ell)$. Thus we have $P(D(G(K)) = 1) = P(\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi_{\mathsf{PRG}}}(\ell) = 1) = \frac{1}{2} + \epsilon(\ell)$.

Therefore

$$|P(D(r) = 1) - P(D(G(K)) = 1)| = \epsilon(\ell)$$

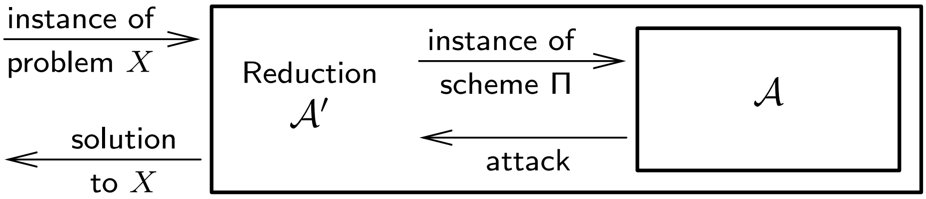
which we have assumed not to be negligible, which implies that G is not a pseudo-random generator, contradicting the assumption. □

Katz/Lindell, pp 73-75

Security proofs through reduction

Some key points about this style of "security proof":

- We have *not* shown that the encryption scheme Π_{PRG} is "secure". (We don't know how to do this!)
- We have shown that Π_{PRG} has one particular type of security property, **if** one of its building blocks (G) has another one.
- We have "reduced" the security of construct Π_{PRG} to another problem X:



instance of problem X

Reduction $\mathcal{A}'$

instance of scheme Π

$\mathcal{A}$

attack

solution to X

Here: X = distinguishing output of G from random string

- We have shown how to turn any successful attack on Π_{PRG} into an equally successful attack on its underlying building block G.
- "Successful attack" means finding a polynomial-time probabilistic adversary algorithm that succeeds with non-negligible success probability in winning the game specified by the given security definition.

Security proofs through reduction

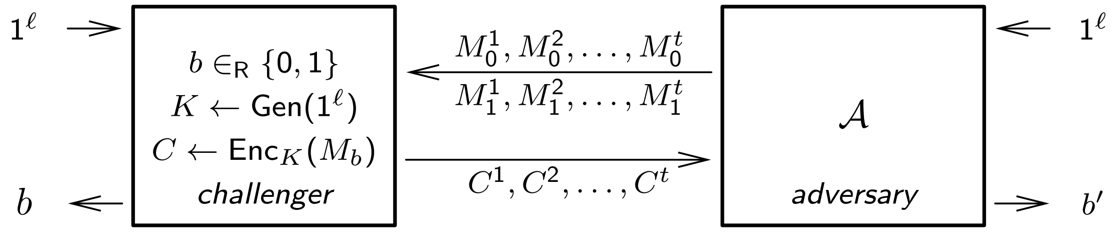
In the end, the provable security of some cryptographic construct (e.g., Π_{PRG}, some mode of operation, some security protocol) boils down to these questions:

- What do we expect from the construct?
- What do we expect from the underlying building blocks?
- Does the construct introduce new weaknesses?
- Does the construct mitigate potential existing weaknesses in its underlying building blocks?

Security for multiple encryptions

Private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathcal{M} = \{0,1\}^m$, security parameter ℓ.

Experiment/game $\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(\ell)$:



1^ℓ

$b \in_{\mathsf{R}} \{0,1\}$

$K \leftarrow \mathsf{Gen}(1^\ell)$

$C \leftarrow \mathsf{Enc}_K(M_b)$

challenger

b

$M_0^1, M_0^2, \ldots, M_0^t$

$M_1^1, M_1^2, \ldots, M_1^t$

$C^1, C^2, \ldots, C^t$

$\mathcal{A}$

adversary

1^ℓ

b'

Setup:

1. The challenger generates a bit $b \in_{\mathsf{R}} \{0,1\}$ and a key $K \leftarrow \mathsf{Gen}(1^\ell)$.
2. The adversary $\mathcal{A}$ is given input 1^ℓ

Rules for the interaction:

1. The adversary $\mathcal{A}$ outputs two sequences of t messages: $M_0^1, M_0^2, \ldots, M_0^t$ and $M_1^1, M_1^2, \ldots, M_1^t$, where all $M_j^i \in \{0,1\}^m$.
2. The challenger computes $C^i \leftarrow \mathsf{Enc}_K(M_b^i)$ and returns $C^1, C^2, \ldots, C^t$ to $\mathcal{A}$

Finally, $\mathcal{A}$ outputs b'. If $b' = b$ then $\mathcal{A}$ has succeeded $\Rightarrow \mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(\ell) = 1$

Security for multiple encryptions (cont'd)

Definition: A private-key encryption scheme Π has *indistinguishable multiple encryptions in the presence of an eavesdropper* if for all probabilistic, polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl, such that

$$P(\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(\ell) = 1) \leq \frac{1}{2} + \mathsf{negl}(\ell)$$

Same definition as for *indistinguishable encryptions in the presence of an eavesdropper*, except for referring to the multi-message eavesdropping experiment $\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A},\Pi}(\ell)$.

Example: Does our stream cipher Π_{PRG} offer indistinguishable *multiple* encryptions in the presence of an eavesdropper?

Adversary $\mathcal{A}_4$ outputs four messages , and

returns $b' = 1$ iff . $P(\mathsf{PrivK}^{\mathsf{mult}}_{\mathcal{A}_4,\Pi_{\mathsf{PRG}}}(\ell) = 1) =$

Actually: Any encryption scheme is going to fail here!

Securing a stream cipher for multiple encryptions I

How can we still use a stream cipher if we want to encrypt multiple messages $M_1, M_2, \ldots, M_t$ using a pseudo-random generator G?

Synchronized mode

Let the PRG run for longer to produce enough output bits for all messages:

$$G(K) = R_1 \| R_2 \| \ldots \| R_t, \qquad C_i = R_i \oplus M_i$$

$\|$ is concatenation of bit strings

- convenient if $M_1, M_2, \ldots, M_t$ all belong to the same communications session and G is of a type that can produce long enough output
- requires preservation of internal state of G across sessions

Securing a stream cipher for multiple encryptions II

Unsynchronized mode

Some PRGs have two separate inputs, a key K and an "initial vector" IV. The private key K remains constant, while IV is freshly chosen at random for each message, and sent along with the message.

$$\text{for each } i\text{:} \qquad IV_i \in_{\mathsf{R}} \{0,1\}^n, \quad C_i := (IV_i, G(K, IV_i) \oplus M_i)$$

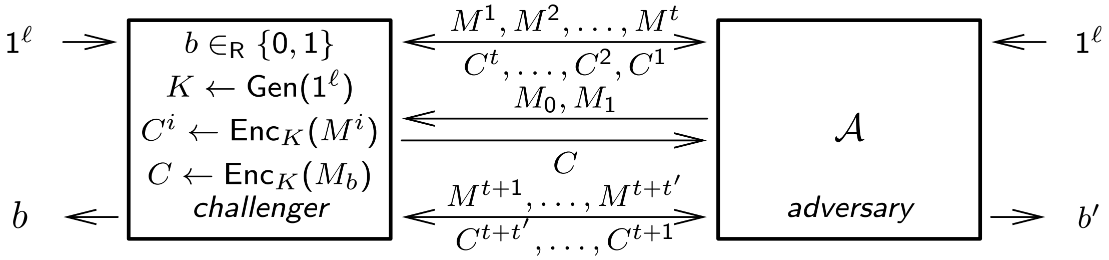
But: what exact security properties do we expect of a G with IV input?

This question leads us to a new security primitive and associated security definition: **pseudo-random functions** and **CPA security**.

Security against chosen-plaintext attacks (CPA)

Private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathcal{M} = \{0,1\}^m$, security parameter ℓ.

Experiment/game $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(\ell)$:



1^ℓ

$b \in_{\mathsf{R}} \{0,1\}$
$K \leftarrow \mathsf{Gen}(1^\ell)$
$C^i \leftarrow \mathsf{Enc}_K(M^i)$
$C \leftarrow \mathsf{Enc}_K(M_b)$
challenger

b

$M^1, M^2, \ldots, M^t$

$C^t, \ldots, C^2, C^1$

M_0, M_1

C

$M^{t+1}, \ldots, M^{t+t'}$

$C^{t+t'}, \ldots, C^{t+1}$

$\mathcal{A}$

adversary

1^ℓ

b'

Setup: (as before)

1. The challenger generates a bit $b \in_{\mathsf{R}} \{0,1\}$ and a key $K \leftarrow \mathsf{Gen}(1^\ell)$.
2. The adversary $\mathcal{A}$ is given input 1^ℓ

Rules for the interaction:

1. The adversary $\mathcal{A}$ is given oracle access to Enc_K:
   $\mathcal{A}$ outputs M^1, gets $\mathsf{Enc}_K(M^1)$, outputs M^2, gets $\mathsf{Enc}_K(M^2)$, ...
2. The adversary $\mathcal{A}$ outputs a pair of messages: $M_0, M_1 \in \{0,1\}^m$.
3. The challenger computes $C \leftarrow \mathsf{Enc}_K(M_b)$ and returns C to $\mathcal{A}$
4. The adversary $\mathcal{A}$ continues to have oracle access to Enc_K.

Finally, $\mathcal{A}$ outputs b'. If $b' = b$ then $\mathcal{A}$ has succeeded $\Rightarrow \mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(\ell) = 1$

Security against chosen-plaintext attacks (cont'd)

Definition: A private-key encryption scheme Π has *indistinguishable multiple encryptions under a chosen-plaintext attack* ("is *CPA-secure*") if for all probabilistic, polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl, such that

$$P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(\ell) = 1) \le \frac{1}{2} + \mathsf{negl}(\ell)$$

Advantages:

- Eavesdroppers can often observe their own text being encrypted, even where the encrypter never intended to provide an oracle. (WW2 story: Midway Island/AF, server communication).
- CPA security provably implies security for multiple encryptions.
- CPA security allows us to build a variable-length encryption scheme simply by using a fixed-length one many times.

Pseudo-random function

$\underset{\text{key}}{F : \{0,1\}^*} \times \underset{\text{input}}{\{0,1\}^*} \rightarrow \underset{\text{output}}{\{0,1\}^*}$ efficient, keyed, $\underset{|\text{input}|=|\text{output}|}{\text{length preserving}}$

Definition

F is a *pseudo-random function* if for all probabilistic, polynomial-time distinguishers D there exists a negligible function negl such that

$$\left| P(D^{F_K(\cdot)}(1^n) = 1) - P(D^{f(\cdot)}(1^n) = 1) \right| \leq \text{negl(n)}$$

where $K \in_\text{R} \{0,1\}^n$ is chosen uniformly at random and f is chosen uniformly at random from the set of functions mapping n-bit strings to n-bitstrings.

Notation: $D^{f(\cdot)}$ means that algorithm D has oracle access to function f.

How does this differ from a pseudo-random generator?

The distinguisher of a pseudo-random generator examines a string. Here, the distinguisher examines entire functions F_K and f.

There are $2^{n \cdot 2^n}$ different functions mapping n-bit strings to n-bit strings, so any description of f would be at least $n \cdot 2^n$ bits long, which cannot be read in polynomial time. Therefore, we need to provide oracle access.

Block ciphers: practical constructions believed to provide pseudo-random functions/permutations.

CPA-secure encryption using a pseudo-random function

We define the following fixed-length private-key encryption scheme:

Π_PRF = (Gen, Enc, Dec):

Let F be a pseudo-random function.

- Gen: on input 1^ℓ choose $K \in_\text{R} \{0,1\}^\ell$ randomly
- Enc: read $K \in \{0,1\}^\ell$ and $M \in \{0,1\}^\ell$, choose $R \in_\text{R} \{0,1\}^\ell$ randomly, then output
$$C := (R, F_K(R) \oplus M)$$
- Dec: read $K \in \{0,1\}^\ell$, $C = (R,S) \in \{0,1\}^{2\ell}$, then output
$$M := F_K(R) \oplus S$$

Strategy for proving Π_PRF to be CPA secure:

1. Show that a variant scheme $\tilde{\Pi}$ in which we replace F_K with a random function f is CPA secure (just not efficient).
2. Show that replacing f with a pseudo-random function F_K cannot make it insecure, by showing how an attacker on the scheme using F_K can be converted into a distinguisher between f and F_K, violating the assumption that F_K is a pseudo-random function.

Security proof for encryption scheme Π_{PRF}

First consider $\tilde{\Pi}$, a variant of Π_{PRF} in which the pseudo-random function F_K was replaced with a random function f. Claim:

$$P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell) = 1) \leq \frac{1}{2} + \frac{q(\ell)}{2^\ell} \qquad \text{with } q(\ell) \text{ oracle queries}$$

Recall: when the challenge ciphertext C in $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell)$ is computed, the challenger picks $R_C \in_{\mathsf{R}} \{0,1\}^\ell$ and returns $C := (R_C, f(R_C) \oplus M_b)$.

Case 1: R_C is also used in one of the oracle queries. In which case $\mathcal{A}$ can easily find out $f(R_C)$ and decrypt M_b. $\mathcal{A}$ makes at most $q(\ell)$ oracle queries and there are 2^ℓ possible values of R_C, this case happens with a probability of at most $q(\ell)/2^\ell$.

Case 2: R_C is not used in any of the oracle queries. For $\mathcal{A}$ the value R_C remains completely random, $f(R_C)$ remains completely random, m_b is returned one-time pad encrypted, and $\mathcal{A}$ can only make a random guess, so in this case $P(b' = b) = \frac{1}{2}$.

$$\begin{aligned} & P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell) = 1) \\ & \quad = P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell) = 1 \wedge \text{Case 1}) + P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell) = 1 \wedge \text{Case 2}) \\ & \quad \leq P(\text{Case 1}) + P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell) = 1 | \text{Case 2}) \leq \frac{q(\ell)}{2^\ell} + \frac{1}{2}. \end{aligned}$$

Security proof for encryption scheme Π_{PRF} (cont'd)

Assume we have an attacker $\mathcal{A}$ against Π_{PRF} with non-negligible

$$\epsilon(\ell) = P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi_{\mathsf{PRF}}}(\ell) = 1) - \frac{1}{2}$$

Its performance against $\tilde{\Pi}$ is also limited by

$$P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell) = 1) \leq \frac{1}{2} + \frac{q(\ell)}{2^\ell}$$

Combining those two equations we get

$$P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi_{\mathsf{PRF}}}(\ell) = 1) - P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell) = 1) \geq \epsilon(\ell) - \frac{q(\ell)}{2^\ell}$$

which is not negligible either, allowing us to distinguish f from F_K:
Build distinguisher $D^{\mathcal{O}}$ using oracle $\mathcal{O}$ to play $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(\ell)$ with $\mathcal{A}$:

1. Run $\mathcal{A}(1^\ell)$ and for each of its oracle queries M^i pick $R^i \in_{\mathsf{R}} \{0,1\}^\ell$, then return $C^i := (R^i, \mathcal{O}(R^i) \oplus M^i)$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs M_0, M_1, pick $b \in_{\mathsf{R}} \{0,1\}$ and $R_C \in_{\mathsf{R}} \{0,1\}^\ell$, then return $C := (R_C, \mathcal{O}(R_C) \oplus M_b)$ to $\mathcal{A}$.
3. Continue answering $\mathcal{A}$'s encryption oracle queries. When $\mathcal{A}$ outputs b', output 1 if $b' = b$, otherwise 0.

Security proof for encryption scheme Π_{PRF} (cont'd)

How effective is this D?

1. **If D's oracle is F_K:** $\mathcal{A}$ effectively plays $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi_{\mathsf{PRF}}}(\ell)$ because if K was chosen randomly, D^{F_K} behaves towards $\mathcal{A}$ just like Π_{PRF}, and therefore

$$P(D^{F_K(\cdot)}(1^\ell) = 1) = P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\Pi_{\mathsf{PRF}}}(\ell) = 1)$$

2. **If D's oracle is f:** likewise, $\mathcal{A}$ effectively plays $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell)$ and therefore

$$P(D^{f(\cdot)}(1^\ell) = 1) = P(\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\Pi}}(\ell) = 1)$$

if $f \in_{\mathsf{R}} (\{0,1\}^\ell)^{\{0,1\}^\ell}$ is chosen uniformly at random.

All combined the difference

$$P(D^{F_K(\cdot)}(1^\ell) = 1) - P(D^{f(\cdot)}(1^\ell) = 1) \geq \epsilon(\ell) - \frac{q(\ell)}{2^\ell}$$

not being negligible implies that F_K is not a pseudo-random function, which contradicts the assumption, so Π_{PRF} is CPA secure. □

Katz/Lindell, pp 90–93

Pseudo-random permutation

$$\underset{\text{key}}{F : \{0,1\}^*} \times \underset{\text{input}}{\{0,1\}^*} \to \underset{\text{output}}{\{0,1\}^*}$$

efficient, keyed, length preserving
|input|=|output|

F_K is a *pseudo-random permutation* if

- for every key K, there is a 1-to-1 relationship for input and output
- F_K and F_K^{-1} can be calculated with polynomial-time algorithms
- there is no polynomial-time distinguisher that can distinguish F_K (with randomly picked K) from a random permutation.

Note: Any pseudo-random permutation is also a pseudo-random function. A random function f looks to any distinguisher just like a random permutation until it finds a collision $x \neq y$ with $f(x) = f(y)$. The probability for finding one in polynomial time is negligible ("birthday problem").

A *strong* pseudo-random permutation remains indistinguishable even if the distinguisher has oracle access to the inverse.

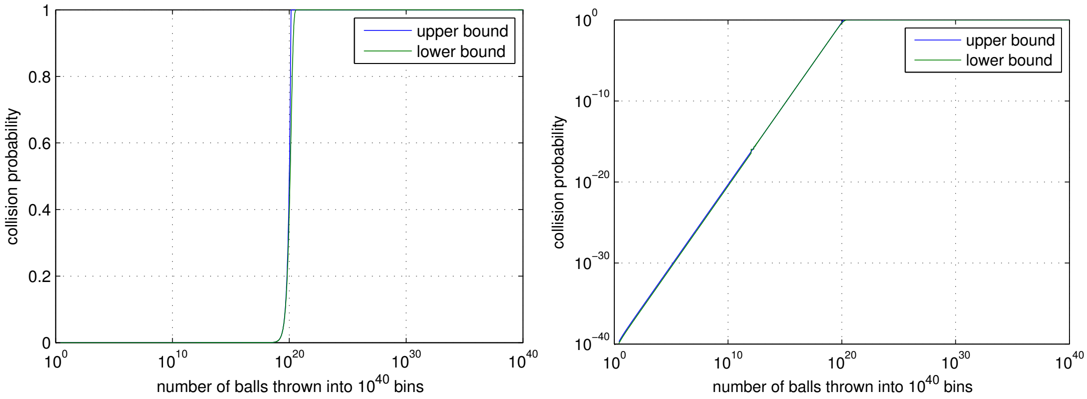
Definition: F is a *strong pseudo-random permutation* if for all polynomial-time distinguishers D there exists a negligible function negl such that

$$\left| P(D^{F_K(\cdot),F_K^{-1}(\cdot)}(1^n) = 1) - P(D^{f(\cdot),f^{-1}(\cdot)}(1^n) = 1) \right| \leq \mathrm{negl}(n)$$

where $K \in_{\mathsf{R}} \{0,1\}^n$ is chosen uniformly at random, and f is chosen uniformly at random from the set of permutations on n-bit strings.

Probability of collision / birthday problem

Throw b balls into n bins, selecting each bin uniformly at random.
With what probability do at least two balls end up in the same bin?



Remember: for large n the collision probability

- is near 1 for $b \gg \sqrt{n}$
- is near 0 for $b \ll \sqrt{n}$, growing roughly proportional to $\frac{b^2}{n}$

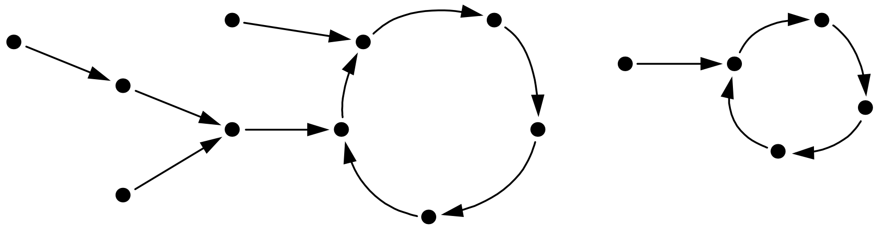
Expected number of balls thrown before first collision: $\sqrt{\frac{\pi}{2}n}$ (for $n \to \infty$)

No simple, efficient, and exact formula for collision probability, but good approximations:
`http://cseweb.ucsd.edu/~mihir/cse207/w-birthday.pdf`

Iterating a random function

$f : \{1, \dots, n\} \to \{1, \dots, n\}$ n^n such functions, pick one at random

Functional graph: vertices $\{1, \dots, n\}$, directed edges $(i, f(i))$



Several components, each a directed cycle and trees attached to it.

Some expected values for $n \to \infty$, random $u \in_{\mathrm{R}} \{1, \dots, n\}$:

- tail length $\mathsf{E}(t(u)) = \sqrt{\pi n/8}$ $f^{t(u)}(u) = f^{t(u)+c(u)\cdot i}(u)$, $\forall i \in \mathbb{N}$,
- cycle length $\mathsf{E}(c(u)) = \sqrt{\pi n/8}$ where $t(u), c(u)$ minimal
- rho-length $\mathsf{E}(t(u) + c(u)) = \sqrt{\pi n/2}$
- predecessors $\mathsf{E}(|\{v|f^i(v) = u \wedge i > 0\}|) = \sqrt{\pi n/8}$
- edges of component containing u: $2n/3$

If f is a random *permutation*: no trees, expected cycle length $(n + 1)/2$

Menezes/van Oorschot/Vanstone, §2.1.6. Knuth: TAOCP, §1.3.3, exercise 17.
Flajolet/Odlyzko: Random mapping statistics, EUROCRYPT'89, LNCS 434.

Modes of operation

Given a fixed-length pseudo-random function F, we could encrypt a variable-length message $M \| \mathsf{Pad}(M) = M_1 \| M_2 \| \dots \| M_n$ by applying Π_{PRF} to its individual blocks M_i, and the result will still be CPA secure:

$$\mathsf{Enc}_K(M) = (R_1, \mathsf{Enc}_K(R_1) \oplus M_1, R_2, \mathsf{Enc}_K(R_2) \oplus M_2, \dots R_n, \mathsf{Enc}_K(R_n) \oplus M_n)$$

But this doubles the message length!

"Modes of operation" that have also been proven to be CPA secure:

Cipher-block chaining (CBC)

$$C_0 \in_{\mathsf{R}} \{0,1\}^m, \quad C_i := G_K(M_i \oplus C_{i-1})$$

Output feedback mode (OFB)

$$C_0 := R_0 \in_{\mathsf{R}} \{0,1\}^m, \quad R_i := G_K(R_{i-1}), \quad C_i := M_i \oplus R_i$$

Randomized counter mode (CNT)

$$C_0 \in_{\mathsf{R}} \{0,1\}^m, \quad C_i := M_i \oplus F_K(C_0 + i)$$

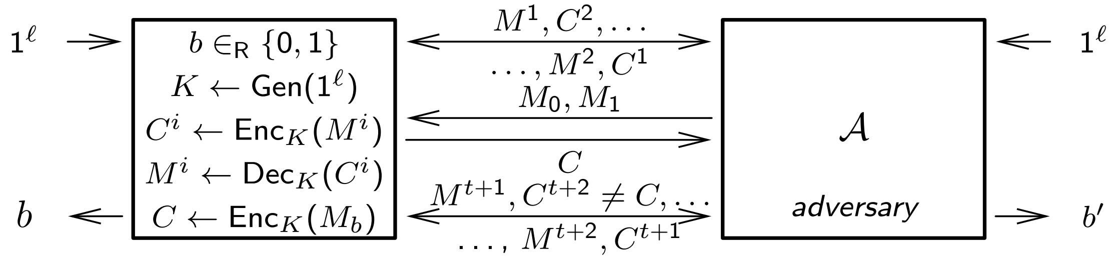
$$\mathsf{Enc}_K(M_1 \| M_2 \| \dots \| M_n) = (C_0 \| C_1 \| C_2 \| \dots \| C_n)$$

Above, F is a pseudo-random function and G is a pseudo-random permutation. The security depends on both their key size and block size.

Security against chosen-ciphertext attacks (CCA)

Private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathcal{M} = \{0,1\}^m$, security parameter ℓ.

Experiment/game $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(\ell)$:



1^ℓ

$b \in_{\mathsf{R}} \{0,1\}$

$K \leftarrow \mathsf{Gen}(1^\ell)$

$C^i \leftarrow \mathsf{Enc}_K(M^i)$

$M^i \leftarrow \mathsf{Dec}_K(C^i)$

$C \leftarrow \mathsf{Enc}_K(M_b)$

b

$M^1, C^2, \dots$

$\dots, M^2, C^1$

M_0, M_1

C

$M^{t+1}, C^{t+2} \neq C, \dots$

$\dots, M^{t+2}, C^{t+1}$

$\mathcal{A}$

adversary

1^ℓ

b'

Setup:

- handling of ℓ, b, K as before

Rules for the interaction:

1. The adversary $\mathcal{A}$ is given oracle access to Enc_K and Dec_K: $\mathcal{A}$ outputs M^1, gets $\mathsf{Enc}_K(M^1)$, outputs C^2, gets $\mathsf{Dec}_K(C^2)$, ...
2. The adversary $\mathcal{A}$ outputs a pair of messages: $M_0, M_1 \in \{0,1\}^m$.
3. The challenger computes $C \leftarrow \mathsf{Enc}_K(M_b)$ and returns C to $\mathcal{A}$
4. The adversary $\mathcal{A}$ continues to have oracle access to Enc_K and Dec_K but is not allowed to ask for $\mathsf{Dec}_K(C)$.

Finally, $\mathcal{A}$ outputs b'. If $b' = b$ then $\mathcal{A}$ has succeeded $\Rightarrow \mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(\ell) = 1$

Malleability

We call an encryption scheme (Gen, Enc, Dec) **malleable** if an adversary can modify the ciphertext in a way that causes a predictable/useful modification to the plaintext.

Example: stream ciphers allow adversary to XOR the plaintext M with arbitrary value X:

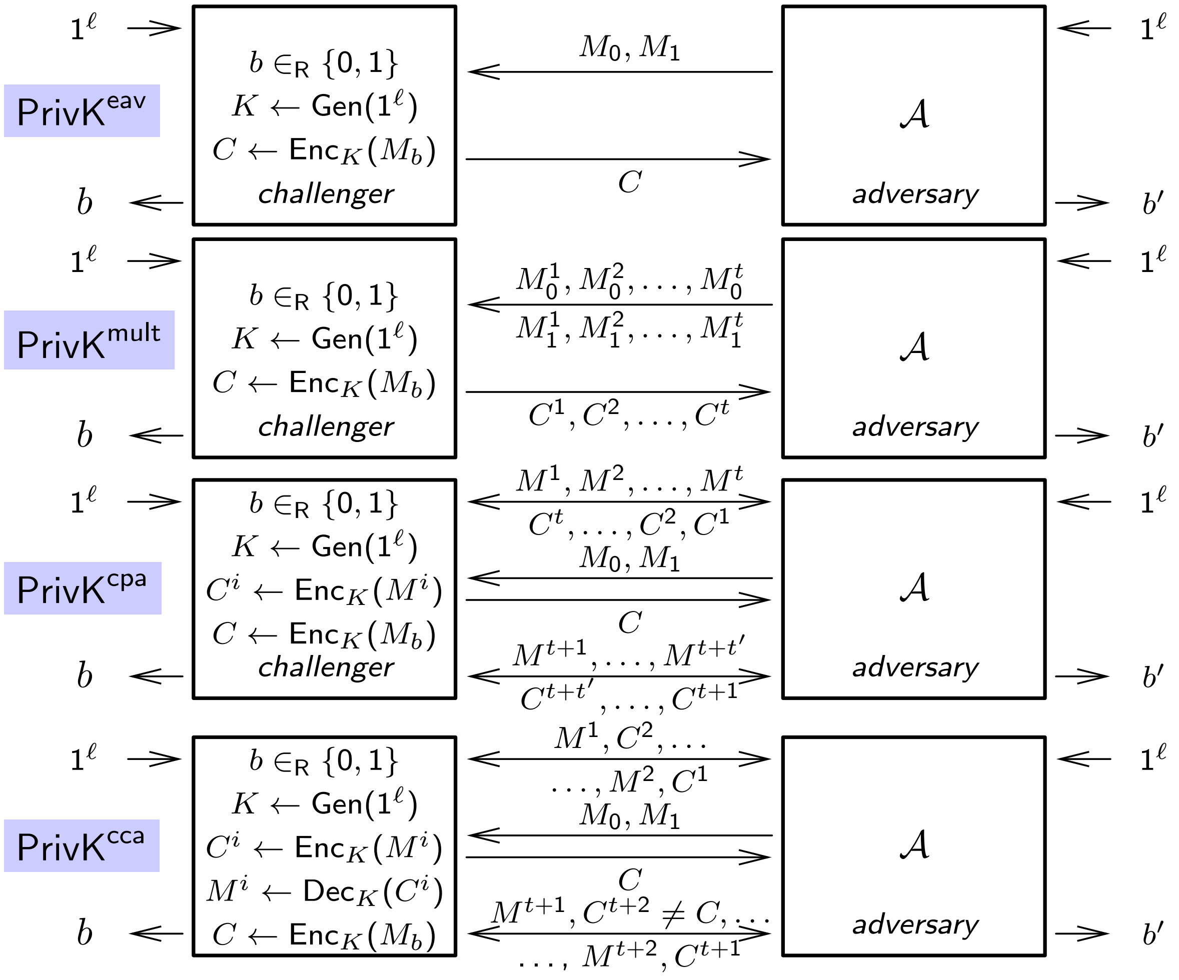
$$\begin{aligned} C &= \mathsf{Enc}_K(M) = (R, F_K(R) \oplus M) \\ C' &= (R, (F_K(R) \oplus M) \oplus X) \\ M' &= \mathsf{Dec}_K(C') = F_K(R) \oplus ((F_K(R) \oplus M) \oplus X) = M \oplus X \end{aligned}$$

Malleable encryption schemes are usually not CCA secure.

CBC, OFB, and CNT are all malleable and not CCA secure.

Malleability is not necessarily a bad thing. If carefully used, it can be an essential building block to privacy-preserving technologies such as digital cash or anonymous electonic voting schemes.

Homomorphic encryption schemes are malleable by design, providing anyone not knowing the key a means to transform the ciphertext of M into a valid encryption of $f(M)$ for some restricted class of transforms f.



PrivK$^{\text{eav}}$

1^ℓ; $b \in_R \{0,1\}$; $K \leftarrow \mathsf{Gen}(1^\ell)$; $C \leftarrow \mathsf{Enc}_K(M_b)$; *challenger*; b

M_0, M_1; C

$\mathcal{A}$; *adversary*; 1^ℓ; b'

PrivK$^{\text{mult}}$

1^ℓ; $b \in_R \{0,1\}$; $K \leftarrow \mathsf{Gen}(1^\ell)$; $C \leftarrow \mathsf{Enc}_K(M_b)$; *challenger*; b

$M_0^1, M_0^2, \dots, M_0^t$; $M_1^1, M_1^2, \dots, M_1^t$; $C^1, C^2, \dots, C^t$

$\mathcal{A}$; *adversary*; 1^ℓ; b'

PrivK$^{\text{cpa}}$

1^ℓ; $b \in_R \{0,1\}$; $K \leftarrow \mathsf{Gen}(1^\ell)$; $C^i \leftarrow \mathsf{Enc}_K(M^i)$; $C \leftarrow \mathsf{Enc}_K(M_b)$; *challenger*; b

$M^1, M^2, \dots, M^t$; $C^t, \dots, C^2, C^1$; M_0, M_1; C; $M^{t+1}, \dots, M^{t+t'}$; $C^{t+t'}, \dots, C^{t+1}$

$\mathcal{A}$; *adversary*; 1^ℓ; b'

PrivK$^{\text{cca}}$

1^ℓ; $b \in_R \{0,1\}$; $K \leftarrow \mathsf{Gen}(1^\ell)$; $C^i \leftarrow \mathsf{Enc}_K(M^i)$; $M^i \leftarrow \mathsf{Dec}_K(C^i)$; $C \leftarrow \mathsf{Enc}_K(M_b)$; b

$M^1, C^2, \dots$; $\dots, M^2, C^1$; M_0, M_1; C; $M^{t+1}, C^{t+2} \neq C, \dots$; $\dots, M^{t+2}, C^{t+1}$

$\mathcal{A}$; *adversary*; 1^ℓ; b'

1. Symmetric encryption
2. **Message authenticity**
3. Authenticated encryption
4. Asymmetric encryption
5. Number theory
6. RSA trapdoor function

Message authentication code (MAC)

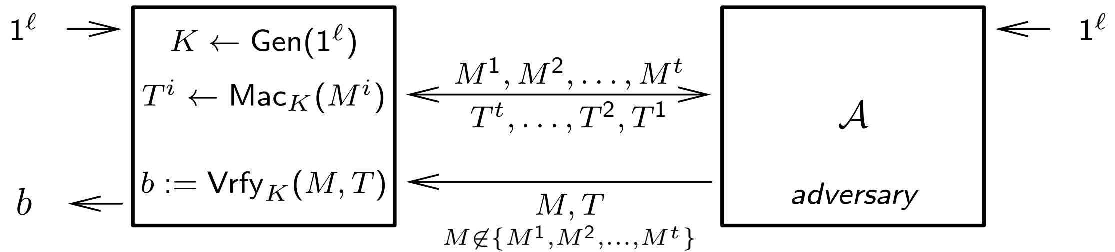
A **message authentication code** is a tuple of probabilistic polynomial-time algorithms (Gen, Mac, Vrfy) and sets $\mathcal{K}, \mathcal{M}$ such that

- the **key generation algorithm** Gen receives a security parameter ℓ and outputs a key $K \leftarrow \mathsf{Gen}(1^\ell)$, with $K \in \mathcal{K}$, key length $|K| \geq \ell$;
- the **tag-generation algorithm** Mac maps a key K and a message $M \in \mathcal{M} = \{0,1\}^*$ to a tag $T \leftarrow \mathsf{Mac}_K(M)$;
- the **verification algorithm** Vrfy maps a key K, a message M and a tag T to an output bit $b := \mathsf{Vrfy}_K(M,T) \in \{0,1\}$, with $b = 1$ meaning the tag is "valid" and $b = 0$ meaning it is "invalid".
- for all ℓ, $K \leftarrow \mathsf{Gen}(1^\ell)$, and $M \in \{0,1\}^m$: $\mathsf{Vrfy}_K(M, \mathsf{Mac}_K(M)) = 1$.

MAC security definition: existential unforgeability

Message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$, $\mathcal{M} = \{0,1\}^*$, security parameter ℓ.

Experiment/game $\text{Mac-forge}_{\mathcal{A},\Pi}(\ell)$:



1^ℓ

$K \leftarrow \mathsf{Gen}(1^\ell)$

$T^i \leftarrow \mathsf{Mac}_K(M^i)$

$b := \mathsf{Vrfy}_K(M,T)$

b

$M^1, M^2, \ldots, M^t$

$T^t, \ldots, T^2, T^1$

M, T

$M \notin \{M^1, M^2, \ldots, M^t\}$

$\mathcal{A}$

adversary

1^ℓ

1. challenger generates random key $K \leftarrow \mathsf{Gen}(1^\ell)$
2. adversary $\mathcal{A}$ is given oracle access to $\mathsf{Mac}_K(\cdot)$; let $\mathcal{Q} = \{M^1, \ldots, M^t\}$ denote the set of queries that $\mathcal{A}$ asks the oracle
3. adversary outputs (M, T)
4. the experiment outputs 1 if $\mathsf{Vrfy}_K(M,T) = 1$ and $M \notin \mathcal{Q}$

Definition: A message authentication code $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is *existentially unforgeable under an adaptive chosen-message attack* ("secure") if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl such that

$$P(\text{Mac-forge}_{\mathcal{A},\Pi}(\ell) = 1) \leq \text{negl}(\ell)$$

MACs versus security protocols

MACs prevent adversaries forging new messages. But adversaries can still

1. replay messages seen previously ("pay £1000", old CCTV image)
2. drop or delay messages ("smartcard revoked")
3. reorder a sequence of messages
4. redirect messages to different recipients

A *security protocol* is a higher-level mechanism that can be built using MACs, to prevent such manipulations. This usually involves including into each message additional data before calculating the MAC, such as

- nonces
  - message sequence counters
  - message timestamps and expiry times
  - random challenge from the recipient
  - MAC of the previous message
- identification of source, destination, purpose, protocol version
- "heartbeat" (regular message to confirm sequence number)

Security protocols also need to define unambiguous syntax for such message fields, delimiting them securely from untrusted payload data.

MAC using a pseudo-random function

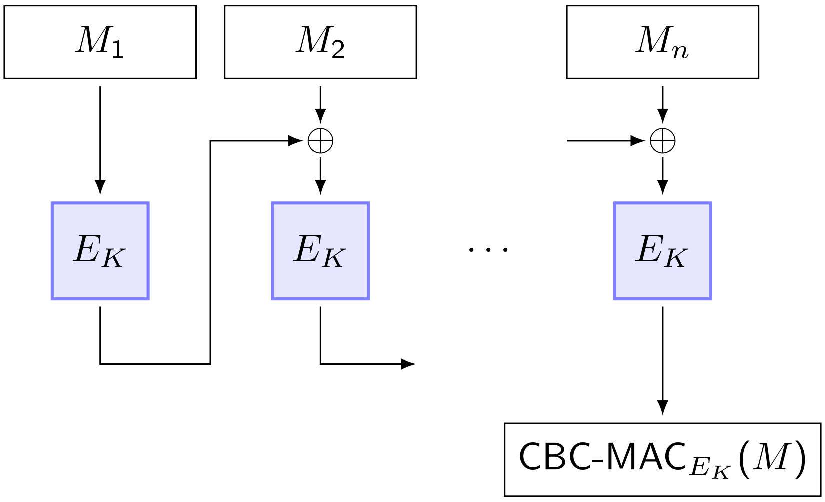
Let F be a pseudo-random function.

- Gen: on input 1^ℓ choose $K \in_\mathsf{R} \{0,1\}^\ell$ randomly
- Mac: read $K \in \{0,1\}^\ell$ and $M \in \{0,1\}^m$, then output $T := F_K(M) \in \{0,1\}^n$
- Vrfy: read $K \in \{0,1\}^\ell$, $M \in \{0,1\}^m$, $T \in \{0,1\}^n$, then output 1 iff $T = F_K(M)$.

If F is a pseudo-random function, then $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is existentially unforgeable under an adaptive chosen message attack.

MAC using a block cipher: CBC-MAC

Blockcipher $E : \{0,1\}^\ell \times \{0,1\}^m \to \{0,1\}^m$



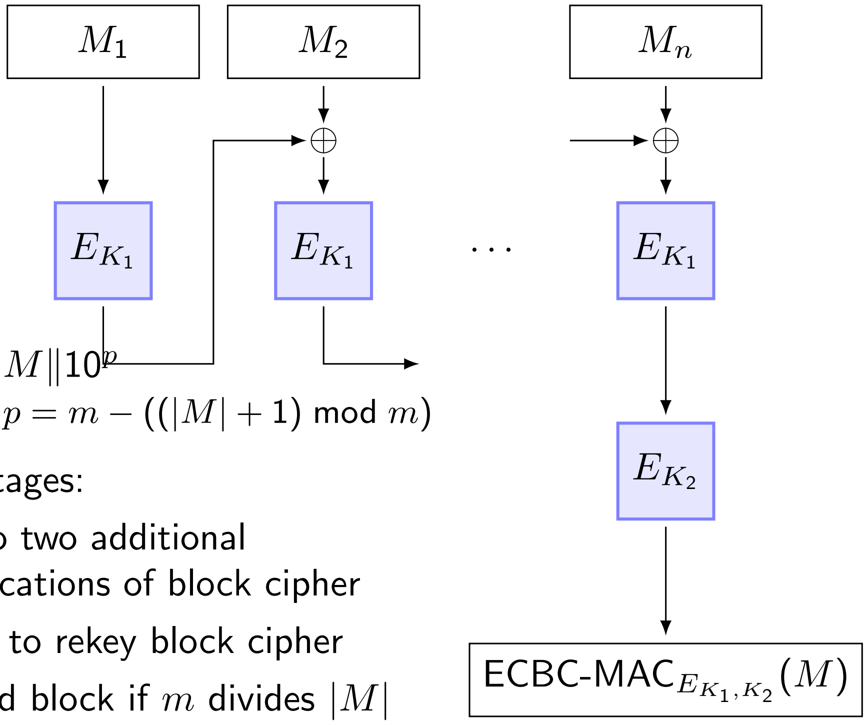
Similar to CBC: $IV = 0^m$, last ciphertext block serves as tag.

Provides existential unforgeability, but only for **fixed** message length n:

Adversary asks oracle for $T^1 :=$ CBC-MAC$_{E_K}(M^1) = E_K(M^1)$ and then presents $M = M^1 \| (T^1 \oplus M^1)$ and $T :=$ CBC-MAC$_{E_K}(M) =$ $E_K((M^1 \oplus T^1) \oplus E_K(M^1)) = E_K((M^1 \oplus T^1) \oplus T^1) = E_K(M^1) = T^1$.

Variable-length MAC using a block cipher: ECBC-MAC

Blockcipher $E : \{0,1\}^\ell \times \{0,1\}^m \rightarrow \{0,1\}^m$



Padding: $M \| 10^p$

$p = m - ((|M| + 1) \bmod m)$

Disadvantages:

- up to two additional applications of block cipher
- need to rekey block cipher
- added block if m divides $|M|$

Variable-length MAC using a block cipher: CMAC

Blockcipher $E : \{0,1\}^\ell \times \{0,1\}^m \rightarrow \{0,1\}^m$ (typically AES: $m = 128$)

Derive subkeys $K_1, K_2 \in \{0,1\}^m$ from key $K \in \{0,1\}^\ell$:

- $K_0 := E_K(0)$
- if $\text{msb}(K_0) = 0$ then $K_1 := (K_0 \ll 1)$ else $K_1 := (K_0 \ll 1) \oplus J$
- if $\text{msb}(K_1) = 0$ then $K_2 := (K_1 \ll 1)$ else $K_2 := (K_1 \ll 1) \oplus J$

This merely clocks a linear-feedback shift register twice, or equivalently multiplies a value in $GF(2^m)$ twice with x. J is a fixed constant (generator polynomial), $\ll$ is a left shift.

CMAC algorithm:

```
$M_1 \| M_2 \| \ldots \| M_n := M$
$r := |M_n|$
**if** $r = m$ **then** $M_n := K_1 \oplus M_n$
**else** $M_n := K_2 \oplus (M_n \| 10^{m-r-1})$
**return** CBC-MAC$_K(M_1 \| M_2 \| \ldots \| M_n)$
```

Provides existential unforgeability, without the disadvantages of ECBC.

NIST SP 800-38B, RFC 4493

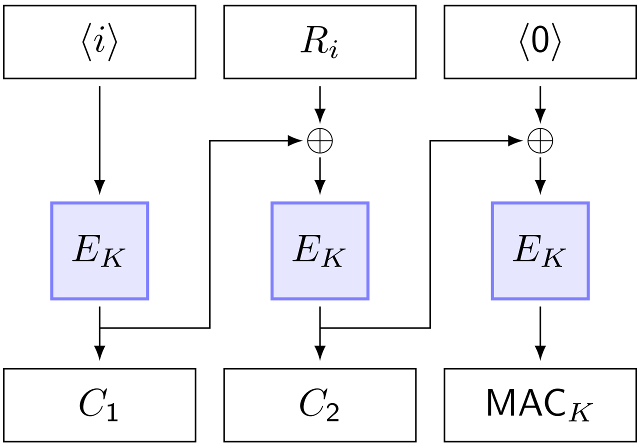
Birthday attack against CBC-MAC, ECBC-MAC, CMAC

Let E be an m-bit block cipher, used to build MAC_K with m-bit tags.

Birthday/collision attack:

- Make $t \approx \sqrt{2^m}$ oracle queries for $T^i := \text{MAC}_K(\langle i \rangle \| R_i \| \langle 0 \rangle)$ with $R_i \in_\text{R} \{0,1\}^m$, $1 \leq i \leq t$.
  Here $\langle i \rangle \in \{0,1\}^m$ is the m-bit binary integer notation for i.
- Look for collision $T^i = T^j$ with $i \neq j$
- Ask oracle for $T' := \text{MAC}_K(\langle i \rangle \| R_i \| \langle 1 \rangle)$
- Present $M := \langle j \rangle \| R_j \| \langle 1 \rangle$ and $T := T' = \text{MAC}_K(M)$

The same intermediate value C_2 occurs while calculating the MAC of
$\langle i \rangle \| R_i \| \langle 0 \rangle$, $\langle j \rangle \| R_j \| \langle 0 \rangle$,
$\langle i \rangle \| R_i \| \langle 1 \rangle$, $\langle j \rangle \| R_j \| \langle 1 \rangle$.



Possible workaround:
Truncate MAC result to less than m bits, such that adversary cannot easily spot collisions in C_2 from C_3.

Solution: big enough m.

1 Symmetric encryption

2 Message authenticity

3 Authenticated encryption

4 Asymmetric encryption

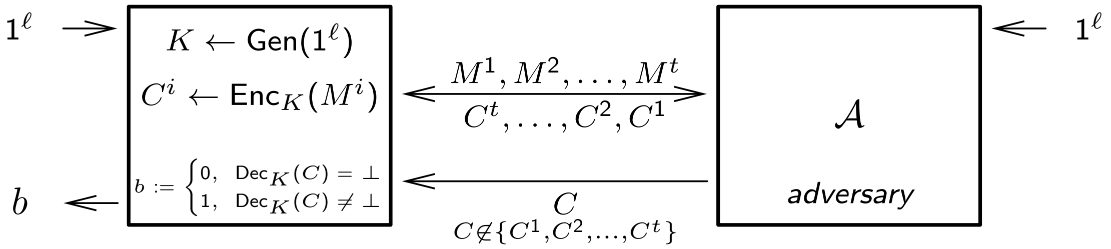
5 Number theory

6 RSA trapdoor function

Ciphertext integrity

Private-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, Dec can output error: $\perp$

Experiment/game $\mathsf{CI}_{\mathcal{A},\Pi}(\ell)$:


1^ℓ
$K \leftarrow \mathsf{Gen}(1^\ell)$
$C^i \leftarrow \mathsf{Enc}_K(M^i)$
$b := \begin{cases} 0, & \mathsf{Dec}_K(C) = \perp \\ 1, & \mathsf{Dec}_K(C) \neq \perp \end{cases}$
b
$M^1, M^2, \ldots, M^t$
$C^t, \ldots, C^2, C^1$
C
$C \notin \{C^1, C^2, \ldots, C^t\}$
$\mathcal{A}$
adversary
1^ℓ

1. challenger generates random key $K \leftarrow \mathsf{Gen}(1^\ell)$
2. adversary $\mathcal{A}$ is given oracle access to $\mathsf{Enc}_K(\cdot)$; let $\mathcal{Q} = \{C^1, \ldots, C^t\}$ denote the set of query answers that $\mathcal{A}$ got from the oracle
3. adversary outputs C
4. the experiment outputs 1 if $\mathsf{Dec}_K(C) \neq \perp$ and $C \notin \mathcal{Q}$

Definition: An encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ provides *ciphertext integrity* if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl such that

$$P(\mathsf{CI}_{\mathcal{A},\Pi}(\ell) = 1) \leq \mathsf{negl}(\ell)$$

Autenticated encryption

Definition: An encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ provides *authenticated encryption* if it provides both CPA security and ciphertext integrity.

Such an encryption scheme will then also be CCA secure.

Example:

Private-key encryption scheme $\Pi_\mathsf{E} = (\mathsf{Gen}_\mathsf{E}, \mathsf{Enc}, \mathsf{Dec})$
Message authentication code $\Pi_\mathsf{M} = (\mathsf{Gen}_\mathsf{M}, \mathsf{Mac}, \mathsf{Vrfy})$

Encryption scheme $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$:

1. $\mathsf{Gen}'(1^\ell) := (K_\mathsf{E}, K_\mathsf{M})$ with $K_\mathsf{E} \leftarrow \mathsf{Gen}_\mathsf{E}(1^\ell)$ and $K_\mathsf{M} \leftarrow \mathsf{Gen}_\mathsf{M}(1^\ell)$
2. $\mathsf{Enc}'_{(K_\mathsf{E},K_\mathsf{M})}(M) := (C, T)$ with $C \leftarrow \mathsf{Enc}_{K_\mathsf{E}}(M)$ and $T \leftarrow \mathsf{Mac}_{K_\mathsf{M}}(C)$
3. Dec' on input of $(K_\mathsf{E}, K_\mathsf{M})$ and (C, T) first check if $\mathsf{Vrfy}_{K_\mathsf{M}}(C, T) = 1$. If yes, output $\mathsf{Dec}_{K_\mathsf{E}}(C)$, if no output $\perp$.

If Π_E is a CPA-secure private-key encryption scheme and Π_M is a secure message authentication code with unique tags, then Π' is a CCA-secure private-key encryption scheme.

A message authentication code has *unique tags*, if for every K and every M there exists a unique value T, such that $\mathsf{Vrfy}_K(M, T) = 1$.

Combining encryption and message authentication

Warning: Not every way of combining a CPA-secure encryption scheme (to achieve privacy) and a secure message authentication code (to prevent forgery) will necessarily provide CPA security:

Encrypt-and-authenticate: $(\mathsf{Enc}_{K_E}(M), \mathsf{Mac}_{K_M}(M))$
Unlikely to be CPA secure: MAC may leak information about M.

Authenticate-then-encrypt: $\mathsf{Enc}_{K_E}(M \| \mathsf{Mac}_{K_M}(M))$
May not be CPA secure: the recipient first decrypts the received message with Dec_{K_E}, then parses the result into M and $\mathsf{Mac}_{K_M}(M)$ and finally tries to verify the latter. A malleable encryption scheme, combined with a parser that reports syntax errors, may reveal information about M.

Encrypt-then-authenticate: $(\mathsf{Enc}_{K_E}(M), \mathsf{Mac}_{K_M}(\mathsf{Enc}_{K_E}(M)))$
Secure: provides both CCA security and existential unforgeability.
If the recipient does not even attempt to decrypt M unless the MAC has been verified successfully, this method can also prevent some side-channel attacks.

Note: CCA security alone does not imply existential unforgeability.

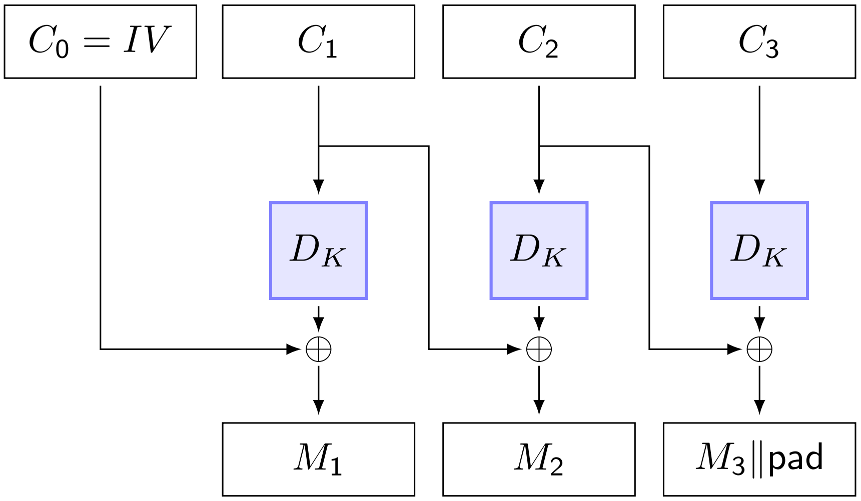
Padding oracle

TLS record protocol:

Recipient steps: CBC decryption, then checks and removes padding, finally checks MAC.

Padding: append n times byte n $(1 \leq n \leq 16)$

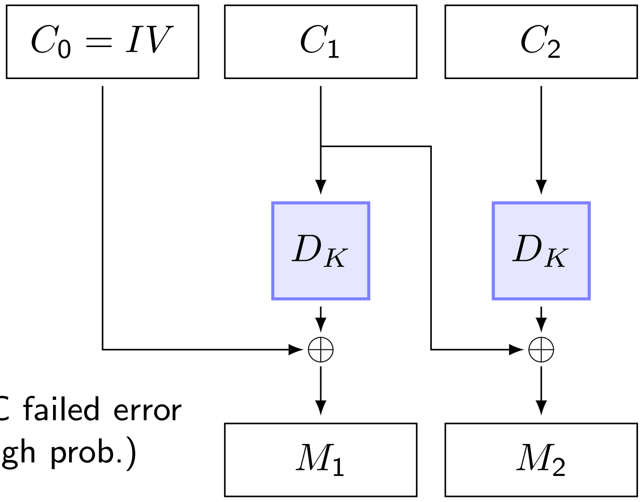
Padding syntax error and MAC failure (used to be) distinguished in error messages.



Padding oracle (cont'd)

Attacker has $C_0, \ldots, C_3$ and tries to get M_2:

- truncate ciphertext after C_2
- $a =$ actual last byte of M_2,
  $g =$ attacker's guess of a
  (try all $g \in \{0, \ldots, 255\}$)
- XOR the last byte of C_1 with
  $g \oplus \texttt{0x01}$
- last byte of M_2 is now
  $a \oplus g \oplus \texttt{0x01}$
- $g = a$: padding correct $\Rightarrow$ MAC failed error
  $g \neq a$: padding syntax error (high prob.)



Then try `0x02 0x02` and so on.

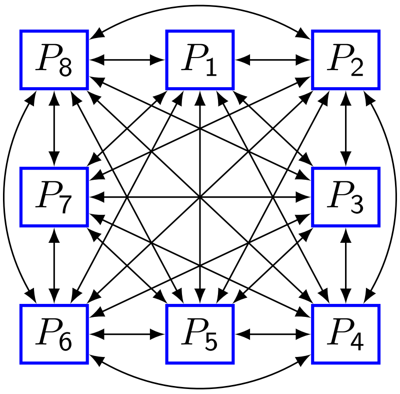
Serge Vaudenay: Security flaws induced by CBC padding, EUROCRYPT 2002

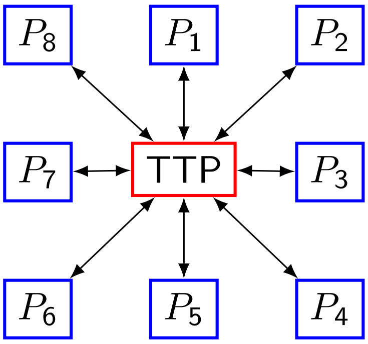
1. Symmetric encryption
2. Message authenticity
3. Authenticated encryption
4. **Asymmetric encryption**
5. Number theory
6. RSA trapdoor function

Key distribution problem

In a group of n participants, there are $n(n-1)/2$ pairs who might want to communicate at some point, requiring $O(n^2)$ keys to be exchanged securely in advance.

This gets quickly unpractical if $n \gg 2$ and if participants regularly join and leave the group.





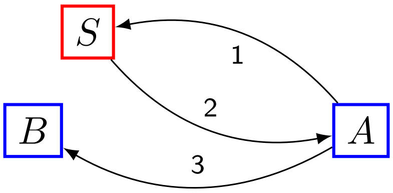
Alternative 1: introduce an intermediary "trusted third party"

Trusted third party – key distribution centre

Needham–Schroeder protocol

Communal trusted server S shares key K_{PS} with each participant P.

1. A informs S that it wants to communicate with B.
2. S generates K_{AB} and replies to A with $\text{Enc}_{K_{AS}}(B, K_{AB}, \text{Enc}_{K_{BS}}(A, K_{AB}))$
   Enc is a symmetric authenticated-encryption scheme
3. A checks name of B, stores K_{AB}, and forwards the "ticket" $\text{Enc}_{K_{BS}}(A, K_{AB})$ to B
4. B also checks name of A and stores K_{AB}.
5. A and B now share K_{AB} and communicate via $\text{Enc}_{K_{AB}}/\text{Dec}_{K_{AB}}$.



Kerberos

An extension of the Needham–Schroeder protocol is now widely used in corporate computer networks between desktop computers and servers, in the form of Kerberos and Microsoft's Active Directory. K_{AS} is generated from A's password (hash function).

Extensions include:

- timestamps and nonces to prevent replay attacks
- a "ticket-granting ticket" is issued and cached at the start of a session, replacing the password for a limited time, allowing the password to be instantly wiped from memory again.
- a pre-authentication step ensures that S does not reply with anything encrypted under K_{AS} unless the sender has demonstrated knowledge of K_{AS}, to hinder offline password guessing.
- mechanisms for forwarding and renewing tickets
- support for a federation of administrative domains ("realms")

Key distribution problem: other options

Alternative 2: hardware security modules + conditional access

1. A trusted third party generates a global key K and embeds it securely in tamper-resistant hardware tokens (e.g., smartcard)
2. Every participant receives such a token, which also knows the identity of its owner and that of any groups they might belong to.
3. Each token offers its holder authenticated encryption operations $\mathrm{Enc}_K(\cdot)$ and $\mathrm{Dec}_K(A, \cdot)$.
4. Each encrypted message $\mathrm{Enc}_K(A, M)$ contains the name of the intended recipient A (or the name of a group to which A belongs).
5. A's smartcard will only decrypt messages addressed this way to A.

Commonly used for "broadcast encryption", e.g. pay-TV, navigation satellites.

Alternative 3: Public-key cryptography

- Find an encryption scheme where separate keys can be used for encryption and decryption.
- Publish the encryption key: the "public key"
- Keep the decryption key: the "secret key"

Some form of trusted third party is usually still required to certify the correctness of the published public keys, but it is no longer directly involved in establishing a secure connection.

Public-key encryption

A **public-key encryption scheme** is a tuple of probabilistic polynomial-time algorithms (Gen, Enc, Dec) such that

- the **key generation algorithm** Gen receives a security parameter ℓ and outputs a pair of keys $(PK, SK) \leftarrow \text{Gen}(1^\ell)$, with key lengths $|PK| \geq \ell$, $|SK| \geq \ell$;
- the **encryption algorithm** Enc maps a public key PK and a plaintext message $M \in \mathcal{M}$ to a ciphertext message $C \leftarrow \text{Enc}_{PK}(M)$;
- the **decryption algorithm** Dec maps a secret key SK and a ciphertext C to a plaintext message $M := \text{Dec}_{SK}(C)$, or outputs $\perp$;
- for all ℓ, $(PK, SK) \leftarrow \text{Gen}(1^\ell)$: $\text{Dec}_{SK}(\text{Enc}_{PK}(M)) = M$.

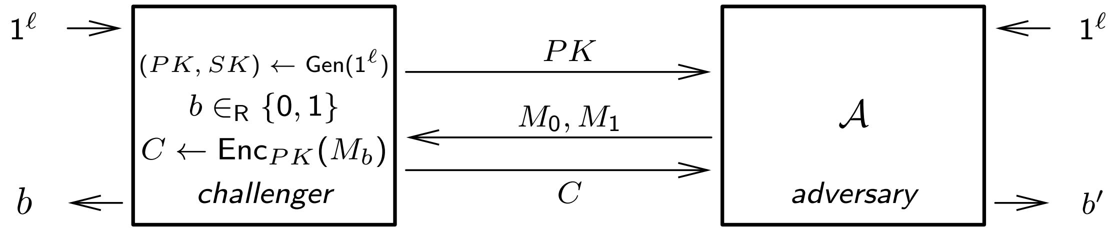
In practice, the message space $\mathcal{M}$ may depend on PK.

In some practical schemes, the condition $\text{Dec}_{SK}(\text{Enc}_{PK}(M)) = M$ may fail with negligible probability.

Security against chosen-plaintext attacks (CPA)

Public-key encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$

Experiment/game $\text{PubK}^{\text{cpa}}_{\mathcal{A},\Pi}(\ell)$:



1^ℓ

$(PK, SK) \leftarrow \text{Gen}(1^\ell)$

$b \in_{\text{R}} \{0, 1\}$

$C \leftarrow \text{Enc}_{PK}(M_b)$

challenger

b

PK

M_0, M_1

C

$\mathcal{A}$

adversary

1^ℓ

b'

Setup:

1. The challenger generates a bit $b \in_{\text{R}} \{0, 1\}$ and a key pair $(PK, SK) \leftarrow \text{Gen}(1^\ell)$.
2. The adversary $\mathcal{A}$ is given input 1^ℓ

Rules for the interaction:

1. The adversary $\mathcal{A}$ is given the public key PK
2. The adversary $\mathcal{A}$ outputs a pair of messages: $M_0, M_1 \in \{0, 1\}^m$.
3. The challenger computes $C \leftarrow \text{Enc}_{PK}(M_b)$ and returns C to $\mathcal{A}$

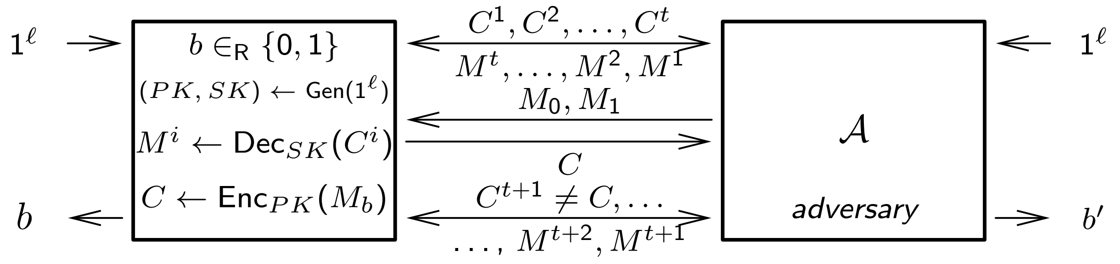
Finally, $\mathcal{A}$ outputs b'. If $b' = b$ then $\mathcal{A}$ has succeeded $\Rightarrow \text{PubK}^{\text{cpa}}_{\mathcal{A},\Pi}(\ell) = 1$

Note that unlike in $\text{PrivK}^{\text{cpa}}$ we do not need to provide $\mathcal{A}$ with any oracle access: here $\mathcal{A}$ has access to the encryption key PK and can evaluate $\text{Enc}_{PK}(\cdot)$ itself.

Security against chosen-ciphertext attacks (CCA)

Public-key encryption scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$

Experiment/game $\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(\ell)$:



1^ℓ

$b \in_{\mathsf{R}} \{0,1\}$

$(PK, SK) \leftarrow \mathsf{Gen}(1^\ell)$

$M^i \leftarrow \mathsf{Dec}_{SK}(C^i)$

$C \leftarrow \mathsf{Enc}_{PK}(M_b)$

b

$C^1, C^2, \ldots, C^t$

$M^t, \ldots, M^2, M^1$

M_0, M_1

C

$C^{t+1} \neq C, \ldots$

$\ldots, M^{t+2}, M^{t+1}$

$\mathcal{A}$

adversary

1^ℓ

b'

Setup:

- handling of ℓ, b, PK, SK as before

Rules for the interaction:

1. The adversary $\mathcal{A}$ is given PK and oracle access to Dec_{SK}:
   $\mathcal{A}$ outputs C^1, gets $\mathsf{Dec}_{SK}(C^1)$, outputs C^2, gets $\mathsf{Dec}_{SK}(C^2)$, ...
2. The adversary $\mathcal{A}$ outputs a pair of messages: $M_0, M_1 \in \{0,1\}^m$.
3. The challenger computes $C \leftarrow \mathsf{Enc}_{SK}(M_b)$ and returns C to $\mathcal{A}$
4. The adversary $\mathcal{A}$ continues to have oracle access to Dec_{SK}
   but is not allowed to ask for $\mathsf{Dec}_{SK}(C)$.

Finally, $\mathcal{A}$ outputs b'. If $b' = b$ then $\mathcal{A}$ has succeeded $\Rightarrow \mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(\ell) = 1$

Security against chosen-plaintext attacks (cont'd)

Definition: A public-key encryption scheme Π has *indistinguishable encryptions under a chosen-plaintext attack* ("is *CPA-secure*") if for all probabilistic, polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl, such that

$$P(\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\Pi}(\ell) = 1) \leq \frac{1}{2} + \mathsf{negl}(\ell)$$

Definition: A public-key encryption scheme Π has *indistinguishable encryptions under a chosen-ciphertext attack* ("is *CCA-secure*") if for all probabilistic, polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl, such that

$$P(\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(\ell) = 1) \leq \frac{1}{2} + \mathsf{negl}(\ell)$$

What about ciphertext integrity / authenticated encryption?

Since the adversary has access to the public encryption key PK, there is no useful equivalent notion of authenticated encryption for a public-key encryption scheme.

1. Symmetric encryption
2. Message authenticity
3. Authenticated encryption
4. Asymmetric encryption
5. **Number theory**
6. RSA trapdoor function

Number theory: basic concepts and notation

Set of integers: $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$

- if there exists $c \in \mathbb{Z}$ such that $ac = b$, we say a divides b, or $a \mid b$
  - if $0 < a$ then a is a "divisor" of b
  - if $1 < a < b$ then a is a "factor" of b
  - if a does not divide b: $a \nmid b$
- if $p > 1$ has no factors (only 1 and p as divisors), it is "prime"
- every integer $n > 1$ has a unique prime factorization $n = \prod_i p_i^{e_i}$
- The modulo operator performs integer division and outputs the remainder:

$$a \bmod b = c \quad \Rightarrow \quad 0 \leq c < b \;\wedge\; \exists d \in \mathbb{Z} : a - db = c$$

Examples: $7 \bmod 5 = 2$, $-1 \bmod 10 = 9$

Greatest common divisor

gcd(a, b) is the largest $c \in \mathbb{Z}$ with $c \mid a$ and $c \mid b$

Examples: gcd$(18, 12) = 6$, gcd$(15, 9) = 3$, gcd$(15, 8) = 1$

- gcd$(a, b) =$ gcd(b, a)
- Euclids algorithm (WLOG $a \geq b > 0$):

$$\text{gcd}(a, b) = \begin{cases} b, & \text{if } b \mid a \\ \text{gcd}(b, a \bmod b), & \text{otherwise} \end{cases}$$

- gcd$(a, b) = 1$ means a and b are "relatively prime"
- for all positive integers a, b, there exist integers x and y such that gcd$(a, b) = ax + by$
- Euclids extended algorithm ($a \geq b > 0$):

$$(\text{gcd}(a, b), x, y) :=$$

$$\text{egcd}(a, b) = \begin{cases} (b, 0, 1), & \text{if } b \mid a \\ (d, y, x - yq), & \text{otherwise,} \\ & \text{with } (d, x, y) := \text{egcd}(b, r), \\ & \text{where } a = qb + r,\ 0 \leq r < b \end{cases}$$

Modular arithmetic

Set of *integers modulo* n: $\mathbb{Z}_n = \{0, 1, \ldots, n - 1\}$

When working in $\mathbb{Z}_n$, we apply after each addition, subtraction, multiplication or exponentiation the modulo n operation.

We add/subtract the integer multiple of n needed to get the result back into $\mathbb{Z}_n$.

Examples in $\mathbb{Z}_5$: $4 + 3 = 2$, $4 \cdot 2 = 3$, $4^2 = 1$

$(\mathbb{Z}_n, +)$ is an abelian group and $(\mathbb{Z}_n, +, \cdot)$ is a commutative ring.

This means: that all the usual rules of arithmetic apply, such as commutativity and associativity.

Example: $a(b + c) = ab + ac = ca + ba$

Modular inversion: division in $\mathbb{Z}_n$

In $\mathbb{Z}_n$, element a has a multiplicative inverse a^{-1} (with $aa^{-1} = 1$) if and only if $\gcd(n, a) = 1$.

In this case, the extended Euclidian algorithm gives us

$$nx + ay = 1$$

and since $nx = 0$ in $\mathbb{Z}_n$ for all x, we have $ay = 1$.

Therefore $y = a^{-1}$ is the inverse needed for dividing by a.

- We call the set of all elements in $\mathbb{Z}_n$ that have an inverse the "multiplicative group" of $\mathbb{Z}_n$:

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid \gcd(n, a) = 1\}$$

- If p is prime, then $\mathbb{Z}_p$ is a (finite) field, that is every element except 0 has a multiplicative inverse:

$$\mathbb{Z}_p^* = \{1, \dots, p-1\}$$

Groups

A **group** $(\mathbb{G}, \bullet)$ is a set $\mathbb{G}$ and an operator $\bullet : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$ that have

closure: $a \bullet b \in \mathbb{G}$ for all $a, b \in \mathbb{G}$

associativity: $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all $a, b, c \in \mathbb{G}$

neutral element: there exists an $e \in \mathbb{G}$ such that for all $a \in \mathbb{G}$: $a \bullet e = e \bullet a = a$

inverse element: for each $a \in \mathbb{G}$ there exists some $b \in \mathbb{G}$ such that $a \bullet b = b \bullet a = e$

If $a \bullet b = b \bullet a$ for all $a, b \in \mathbb{G}$, the group is called **commutative** (or **abelian**).

A *subgroup* $\mathbb{H}$ of $\mathbb{G}$ is a subset $\mathbb{H} \subset \mathbb{G}$ that is also a group (same operator $\bullet$).

Alternative notations:

"Additive" group: think of group operator as a kind of "+"

- write 0 for the neutral element and $-g$ for the inverse of $g \in \mathbb{G}$.
- write $g \cdot i := \underbrace{g \bullet g \bullet \cdots \bullet g}_{i \text{ times}}$ $(g \in \mathbb{G}, i \in \mathbb{Z})$

"Multiplicative" group: think of group operator as a kind of "×"

- write 1 for the neutral element and g^{-1} for the inverse of $g \in \mathbb{G}$.
- write $g^i := \underbrace{g \bullet g \bullet \cdots \bullet g}_{i \text{ times}}$ $(g \in \mathbb{G}, i \in \mathbb{Z})$

Finite groups

Let $(\mathbb{G}, \bullet)$ be a group with a finite number of elements $|\mathbb{G}|$.

Practical examples here: $(\mathbb{Z}_n, +)$, $(\mathbb{Z}_n^*, \cdot)$, $(GF(2^n), \oplus)$, $(GF(2^n) \setminus \{0\}, \otimes)$

Terminology:

- The *order of a group* $\mathbb{G}$ is its size $|\mathbb{G}|$
- *order of group element* g in $\mathbb{G}$ is $\mathrm{ord}_{\mathbb{G}}(g) = \min\{i > 0 \mid g^i = 1\}$.

Related notion: the *characteristic of a ring* is the order of 1 in its additive group, i.e. the smallest i with $\underbrace{1 + 1 + \cdots + 1}_{i \text{ times}} = 0$.

Useful facts regarding any element $g \in \mathbb{G}$ in a group of order $m = |\mathbb{G}|$:

- $g^m = 1$, $g^i = g^{i \bmod m}$
- $g^i = g^{i \bmod \mathrm{ord}(g)}$
- $g^x = g^y \Leftrightarrow x \equiv y \pmod{\mathrm{ord}(g)}$
- $\mathrm{ord}(g) \mid m$ "Lagrange's theorem"
- if $\gcd(e, m) = 1$ then $g \mapsto g^e$ is a permutation, and $g \mapsto g^d$ its inverse (i.e., $g^{ed} = g$) if $ed \bmod m = 1$

Proofs: Katz/Lindell, sections 7.1 and 7.3

Cyclic groups

Let $\mathbb{G}$ be a finite (multiplicative) group of order $m = |\mathbb{G}|$.

For $g \in \mathbb{G}$ consider the set

$$\langle g \rangle := \{g^0, g^1, g^2, \ldots\}$$

Note that $|\langle g \rangle| = \mathrm{ord}(g)$ and $\langle g \rangle = \{g^0, g^1, g^2, \ldots, g^{\mathrm{ord}(g)-1}\}$.

Definitions:

- We call g a *generator* of $\mathbb{G}$ if $\langle g \rangle = \mathbb{G}$.
- We call $\mathbb{G}$ *cyclic* if it has a generator.

Useful facts:

- Every cyclic group of order m is isomorphic to $(\mathbb{Z}_m, +)$. $(g^i \mapsto i)$
- $\langle g \rangle$ is a subgroup of $\mathbb{G}$ (subset, a group under the same operator)
- If $|\mathbb{G}|$ is prime, then $\mathbb{G}$ is cyclic and all $g \in \mathbb{G} \setminus \{1\}$ are generators.

  Recall that $\mathrm{ord}(g) \mid |\mathbb{G}|$. We have $\mathrm{ord}(g) \in \{1, |\mathbb{G}|\}$ if $|\mathbb{G}|$ is prime, which makes g either 1 or a generator.

Proofs: Katz/Lindell, sections 7.3

How to find a generator?

Let $\mathbb{G}$ be a cyclic (multiplicative) group of order $m = |\mathbb{G}|$.

- If m is prime, any non-neutral element is a generator. Done.
  But $|\mathbb{Z}_p^*| = p - 1$ is not prime (for $p > 3$)!
- Directly testing for $|\langle g \rangle| \overset{?}{=} m$ is infeasibe for crypto-sized m.
- Fast test: if $m = \prod_i p_i^{e^i}$ is composite, then $g \in \mathbb{G}$ is a generator if and only if $g^{m/p_i} \neq 1$ for all i.
- Sampling a polynomial number of elements of $\mathbb{G}$ for the above test will lead to a generator in polynomial time (of $\log_2 m$) with all but negligible probability.

⇒ Make sure you pick a group of an order with known prime factors.

One possibility:

- Chose a "strong prime" $p = 2q + 1$, where q is also prime
  $\Rightarrow |\mathbb{Z}_p^*| = p - 1 = 2q$ has prime factors 2 and q.

$(\mathbb{Z}_p, +)$ is a cyclic group

For every prime p every element $g \in \mathbb{Z}_p \setminus \{0\}$ is a generator:

$$\mathbb{Z}_p = \langle g \rangle = \{g \cdot i \bmod p \,|\, 0 \leq i \leq p - 1\}$$

Note that this follows from the last fact on slide 70: $\mathbb{Z}_p$ is of order p, which is prime.

Example in $\mathbb{Z}_7$:

$$(1\cdot 0, 1\cdot 1, 1\cdot 2, 1\cdot 2, 1\cdot 4, 1\cdot 5, 1\cdot 6) = (0,1,2,3,4,5,6)$$
$$(2\cdot 0, 2\cdot 1, 2\cdot 2, 2\cdot 2, 2\cdot 4, 2\cdot 5, 2\cdot 6) = (0,2,4,6,1,3,5)$$
$$(3\cdot 0, 3\cdot 1, 3\cdot 2, 3\cdot 2, 3\cdot 4, 3\cdot 5, 3\cdot 6) = (0,3,6,2,5,1,4)$$
$$(4\cdot 0, 4\cdot 1, 4\cdot 2, 4\cdot 2, 4\cdot 4, 4\cdot 5, 4\cdot 6) = (0,4,1,5,2,6,3)$$
$$(5\cdot 0, 5\cdot 1, 5\cdot 2, 5\cdot 2, 5\cdot 4, 5\cdot 5, 5\cdot 6) = (0,5,3,1,6,4,2)$$
$$(6\cdot 0, 6\cdot 1, 6\cdot 2, 6\cdot 2, 6\cdot 4, 6\cdot 5, 6\cdot 6) = (0,6,5,4,3,2,1)$$

- All the non-zero elements of $\mathbb{Z}_7$ are generators
- $\text{ord}(0) = 1$, $\text{ord}(1) = \text{ord}(2) = \text{ord}(3) = \text{ord}(4) = \text{ord}(5) = \text{ord}(6) = 7$

$(\mathbb{Z}_p^*, \cdot)$ is a cyclic group

For every prime p there exists a *generator* $g \in \mathbb{Z}_p^*$ such that

$$\mathbb{Z}_p^* = \{g^i \bmod p \,|\, 0 \leq i \leq p-2\}$$

Note that this does **not** follow from the last fact on slide 70: $\mathbb{Z}_p^*$ is of order $p-1$, which is usually even, not prime.

Example in $\mathbb{Z}_7^*$:

$$\begin{aligned}
(1^0, 1^1, 1^2, 1^3, 1^4, 1^5) &= (1, 1, 1, 1, 1, 1)\\
(2^0, 2^1, 2^2, 2^3, 2^4, 2^5) &= (1, 2, 4, 1, 2, 4)\\
(3^0, 3^1, 3^2, 3^3, 3^4, 3^5) &= (1, 3, 2, 6, 4, 5)\\
(4^0, 4^1, 4^2, 4^3, 4^4, 4^5) &= (1, 4, 2, 1, 4, 2)\\
(5^0, 5^1, 5^2, 5^3, 5^4, 5^5) &= (1, 5, 4, 6, 2, 3)\\
(6^0, 6^1, 6^2, 6^3, 6^4, 6^5) &= (1, 6, 1, 6, 1, 6)
\end{aligned}$$

- 3 and 5 are generators of $\mathbb{Z}_7^*$

  Fast generator test (p. 71), using $|\mathbb{Z}_7^*| = 6 = 2 \cdot 3$: $3^{6/2} = 6, 3^{6/3} = 2, 5^{6/2} = 6, 5^{6/3} = 4$, all $\neq 1$.
- 1, 2, 4, 6 generate *subgroups* of $\mathbb{Z}_7^*$: $\{1\}$, $\{1, 2, 4\}$, $\{1, 2, 4\}$, $\{1, 6\}$
- $\text{ord}(1) = 1$, $\text{ord}(2) = 3$, $\text{ord}(3) = 6$, $\text{ord}(4) = 3$, $\text{ord}(5) = 6$, $\text{ord}(6) = 2$

  The *order* of g in $\mathbb{Z}_p^*$ is the size of the subgroup $\langle g \rangle$.
  Lagrange's theorem: $\text{ord}_{\mathbb{Z}_p^*}(g) \mid p-1$ for all $g \in \mathbb{Z}_p^*$

Fermat's and Euler's theorem

Fermat's little theorem: (1640)

$$p \text{ prime and } \gcd(a, p) = 1 \quad \Rightarrow \quad a^{p-1} \bmod p = 1$$

Euler's phi function:

$$\varphi(n) = |\mathbb{Z}_n^*| = |\{a \in \mathbb{Z}_n \mid \gcd(n, a) = 1\}|$$

- Example: $\varphi(12) = |\{1, 5, 7, 11\}| = 4$
- primes p, q:

$$\begin{aligned}
\varphi(p) &= p - 1\\
\varphi(p^k) &= p^{k-1}(p-1)\\
\varphi(pq) &= (p-1)(q-1)
\end{aligned}$$

- $\gcd(a, b) = 1 \;\Rightarrow\; \varphi(ab) = \varphi(a)\varphi(b)$

Euler's theorem: (1763)

$$\gcd(a, n) = 1 \quad \Leftrightarrow \quad a^{\varphi(n)} \bmod n = 1$$

- this implies that in $\mathbb{Z}_n$: $a^x = a^{x \bmod \varphi(n)}$ for any $a \in \mathbb{Z}_n, x \in \mathbb{Z}$

Chinese remainder theorem

Definition: Let $(\mathbb{G}, \bullet)$ and $(\mathbb{H}, \circ)$ be two groups. A function $f : \mathbb{G} \to \mathbb{H}$ is an *isomorphism* from $\mathbb{G}$ to $\mathbb{H}$ if

- f is a 1-to-1 mapping (bijection)
- $f(g_1 \bullet g_2) = f(g_1) \circ f(g_2)$ for all $g_1, g_2 \in \mathbb{G}$

Chinese remainder theorem:
For any p, q with $\gcd(p, q) = 1$ and $n = pq$, the mapping

$$f : \mathbb{Z}_n \leftrightarrow \mathbb{Z}_p \times \mathbb{Z}_q \qquad f(x) = (x \bmod p, x \bmod q)$$

is an isomorphism, both from $\mathbb{Z}_n$ to $\mathbb{Z}_p \times \mathbb{Z}_q$ and from $\mathbb{Z}_n^*$ to $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$.

Inverse: To get back from $x_p = x \bmod p$ and $x_q = x \bmod q$ to x, we first use Euclid's extended algorithm to find a, b such that $ap + bq = 1$, and then $x = (x_p bq + x_q ap) \bmod n$.

Application: arithmetic operations on $\mathbb{Z}_n$ can instead be done on both $\mathbb{Z}_p$ and $\mathbb{Z}_q$ after this mapping, which may be faster.

Taking roots in $\mathbb{Z}_p$

If $x^e = c$ in $\mathbb{Z}_p$, then x is the "e^{th} root" of c, or $x = c^{1/e}$.

Case 1: $\gcd(e, p-1) = 1$
Find d with $de = 1$ in $\mathbb{Z}_{p-1}$ (Euclid's extended), then $c^{1/e} = c^d$ in $\mathbb{Z}_p$.
Proof: $(c^d)^e = c^{de} = c^{de \bmod \varphi(p)} = c^{de \bmod p-1} = c^1 = c$.

Case 2: $e = 2$ (taking square roots)
$\gcd(2, p-1) \neq 1$ if p odd prime $\Rightarrow$ Euclid's extended alg. no help here.

Quadratic residues
In $\mathbb{Z}_p^*$, $x \mapsto x^2$ is a 2-to-1 function: $x^2 = (-x)^2$.
Example in $\mathbb{Z}_7^*$: $(1^2, 2^2, 3^2, 4^2, 5^2, 6^2) = (1, 4, 2, 2, 4, 1)$

If x has a square root in $\mathbb{Z}_p$, x is a "quadratic residue".
Example: $\mathbb{Z}_7$ has 4 quadratic residues: $\{0, 1, 2, 4\}$.

If p is an odd prime: $\mathbb{Z}_p$ has $(p-1)/2 + 1$ quadratic residues.

Euler's criterion:

$$c^{(p-1)/2} \bmod p = 1 \quad \Leftrightarrow \quad c \text{ is a quadratic residue in } \mathbb{Z}_p^*$$

Example in $\mathbb{Z}_7$: $(7-1)/2 = 3, \quad (1^3, 2^3, 3^3, 4^3, 5^3, 6^3) = (1, 1, 6, 1, 6, 6)$

$c^{(p-1)/2}$ is also called the *Legendre symbol*

Taking square roots in $\mathbb{Z}_p$

If $p \bmod 4 = 3$ and $c \in \mathbb{Z}_p^*$ is a quadratic residue: $\sqrt{c} = c^{(p+1)/4}$ in $\mathbb{Z}_p$.
Proof: $\left[c^{(p+1)/4}\right]^2 = c^{(p+1)/2} = \underbrace{c^{(p-1)/2}}_{=1} \cdot c = c.$

If $p \bmod 4 = 1$ this can also be done efficiently (details omitted here).

Application: solve $ax^2 + bx + c = 0$ in $\mathbb{Z}_p$

Solution: $x = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Algorithms: $\sqrt{b^2 - 4ac}$ as above, $(2a)^{-1}$ using Euclid's extended

Taking roots in $\mathbb{Z}_n$

If n is composite, then we know how to test whether $c^{1/e}$ exists, and how to compute it efficiently, **only** if we know the prime factors of n.

Working in subgroups of $\mathbb{Z}_p^*$

How can we construct a cyclic finite group $\mathbb{G}$ where all non-neutral elements are generators?

Recall that $\mathbb{Z}_p$ has $(p-1)/2 + 1$ quadratic residues. That includes 0, so: $\mathbb{Z}_p^*$ has $q = (p-1)/2$ quadratic residues, exactly half of its elements.
Quadratic residue: an element that is the square of some other element.

Choose p to be a *strong prime*, that is where q is also prime.

Let $\mathbb{G} = \{g^2 \mid g \in \mathbb{Z}_p^*\}$ be the set of quadratic residues of $\mathbb{Z}_p^*$. $\mathbb{G}$ with operator "multiplication mod p" is a subgroup of $\mathbb{Z}_p^*$, with order $|\mathbb{G}| = q$.

Since $\mathbb{G}$ has prime order $|\mathbb{G}| = q$: for all $g \in \mathbb{G} \setminus \{1\}$: $\langle g \rangle = \mathbb{G}$.

```
GENERATE_GROUP($1^\ell$):
  $p \in_R \{(\ell+1)\text{-bit strong primes}\}$
  $q := (p-1)/2$
  $x \in_R \mathbb{Z}_p^* \setminus \{-1, 1\}$
  $g := x^2 \bmod p$
  return $p$, $q$, $g$
```

This technique is widely used to obtain a cyclic finite group of order q and associated generator g for which the Discrete Logarithm Problem and the Decision Diffie–Hellmann Problem are believed to be hard.

Modular exponentiation

In cyclic group $(\mathbb{G}, \bullet)$ (e.g., $\mathbb{G} = \mathbb{Z}_p^*$):
How do we calculate g^e efficiently? $(g \in \mathbb{G},\ e \in \mathbb{N})$

Naive algorithm: $g^e = \underbrace{g \bullet g \bullet \cdots \bullet g}_{e \text{ times}}$

Far too slow for crypto-size e (e.g., $e \approx 2^{128}$)!

Square and multiply algorithm:

Binary representation: $e = \sum_{i=0}^{n} e_i \cdot 2^i, \quad n = \lfloor \log_2 e \rfloor, \quad e_i = \left\lfloor \frac{e}{2^i} \right\rfloor \bmod 2$

Computation:

$$g^{2^0} := g, \quad g^{2^i} := \left(g^{2^{i-1}}\right)^2$$

$$g^e := \prod_{i=0}^{n} \left(g^{2^i}\right)^{e_i}$$

```
SQUARE_AND_MULTIPLY($g, e$):
  $a := g$
  $b := 1$
  for $i := 0$ to $n$ do
    if $\lfloor e/2^i \rfloor \bmod 2 = 1$ then
      $b := b \bullet a$        ← multiply
    $a := a \bullet a$          ← square
  return $b$
```

Side-channel vulnerability: the **if** statement leaks the binary representation of e. "Montgomery's ladder" is an alternative algorithm with fixed control flow.

Number theory: easy and difficult problems

Easy:

- given composite n and $x \in \mathbb{Z}_n^*$: find $x^{-1} \in \mathbb{Z}_n^*$
- given prime p and polynomial $f(x) \in \mathbb{Z}_p[x]$:
  find $x \in \mathbb{Z}_p$ with $f(x) = 0$
  runtime grows linearly with the degree of the polynomial

Difficult:

- given prime p, generator $g \in \mathbb{Z}_p^*$:
  - given value $a \in \mathbb{Z}_p^*$: find x such that $a = g^x$.
    → Discrete Logarithm Problem
  - given values $g^x, g^y \in \mathbb{Z}_p^*$: find g^{xy}.
    → Computational Diffie–Hellman Problem
  - given values $g^x, g^y, z \in \mathbb{Z}_p^*$: tell whether $z = g^{xy}$.
    → Decision Diffie–Hellman Problem
- given a random $n = p \cdot q$, where p and q are ℓ-bit primes ($\ell \geq 1024$):
  - find integers p and q such that $n = p \cdot q$ in $\mathbb{N}$
    → Factoring Problem
  - given a polynomial $f(x)$ of degree > 1:
    find $x \in \mathbb{Z}_n$ such that $f(x) = 0$ in $\mathbb{Z}_n$

Comparison of difficulty

symmetric key	factoring $n = pq$	DH in $\mathbb{Z}_p^*$	DH in EC
80 bits	1024 bits	1024 bits	160 bits
128 bits	3072 bits	3072 bits	256 bits
256 bits	15360 bits	15360 bits	512 bits

1. Symmetric encryption
2. Message authenticity
3. Authenticated encryption
4. Asymmetric encryption
5. Number theory
6. **RSA trapdoor function**

Trapdoor permutations

A **trapdoor permutation** is a tuple of polynomial-time algorithms $(\mathsf{Gen}, F, F^{-1})$ such that

- the **key generation algorithm** Gen receives a security parameter ℓ and outputs a pair of keys $(PK, SK) \leftarrow \mathsf{Gen}(1^\ell)$, with key lengths $|PK| \geq \ell$, $|SK| \geq \ell$;
- the **sampling function** F maps a public key PK and a value $x \in \mathcal{X}$ to a value $y := F_{PK}(x) \in \mathcal{X}$;
- the **inverting function** F^{-1} maps a secret key SK and a value $y \in \mathcal{X}$ to a value $x := F^{-1}_{SK}(y) \in \mathcal{X}$;
- for all ℓ, $(PK, SK) \leftarrow \mathsf{Gen}(1^\ell)$, $x \in \mathcal{X}$: $F^{-1}_{SK}(F_{PK}(x)) = x$.

In practice, the domain $\mathcal{X}$ may depend on PK.

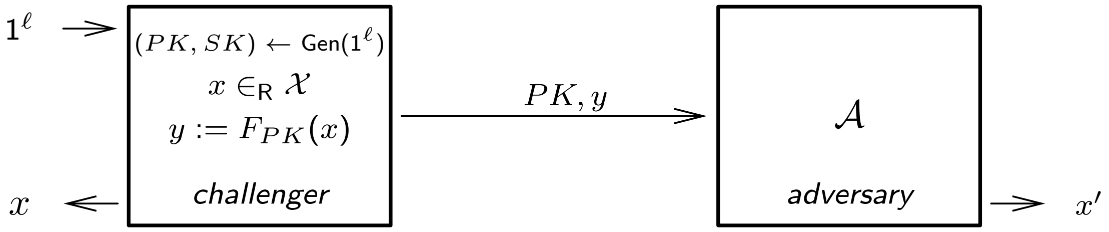
This looks almost like the definition of a public-key encryption scheme, the difference being

- F is deterministic;
- the associated security definition.

Secure trapdoor permutations

Trapdoor permutation: $\Pi = (\mathsf{Gen}, F, F^{-1})$

Experiment/game $\mathsf{TDInv}_{\mathcal{A},\Pi}(\ell)$:


1^ℓ
$(PK, SK) \leftarrow \mathsf{Gen}(1^\ell)$
$x \in_{\mathsf{R}} \mathcal{X}$
$y := F_{PK}(x)$
challenger
x
PK, y
$\mathcal{A}$
adversary
x'

1. The challenger generates a key pair $(PK, SK) \leftarrow \mathsf{Gen}(1^\ell)$ and a random value $x \in_{\mathsf{R}} \mathcal{X}$ from the domain of F_{PK}.
2. The adversary $\mathcal{A}$ is given inputs PK and $y := F_{PK}(x)$.
3. Finally, $\mathcal{A}$ outputs x'.

If $x' = x$ then $\mathcal{A}$ has succeeded: $\mathsf{TDInv}_{\mathcal{A},\Pi}(\ell) = 1$.

A trapdoor permutation Π is secure if for all probabilistic polynomial time adversaries $\mathcal{A}$ the probability of success $P(\mathsf{TDInv}_{\mathcal{A},\Pi}(\ell) = 1)$ is negligible.

While the definition of a trapdoor permutation resembles that of a public-key encryption scheme, its security definition does not provide the adversary any control over the input (plaintext).

Public-key encryption scheme from trapdoor permutation

Trapdoor permutation: $\Pi_{\mathsf{TD}} = (\mathsf{Gen}_{\mathsf{TD}}, F, F^{-1})$ with $F_{PK} : \mathcal{X} \leftrightarrow \mathcal{X}$
Authentic. encrypt. scheme: $\Pi_{\mathsf{AE}} = (\mathsf{Gen}_{\mathsf{AE}}, \mathsf{Enc}, \mathsf{Dec})$, key space $\mathcal{K}$
Secure hash function $h : \mathcal{X} \rightarrow \mathcal{K}$

We define the private-key encryption scheme $\Pi = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$:

- Gen': output key pair $(PK, SK) \leftarrow \mathsf{Gen}_{\mathsf{TD}}(1^{\ell})$
- Enc': on input of plaintext message M, generate random $x \in_{\mathsf{R}} \mathcal{X}$, $y = F(x)$, $K = h(x)$, $C \leftarrow \mathsf{Enc}_K(M)$, output ciphertext (y, C);
- Dec': on input of ciphertext message $C = (y, C)$, recover $K = h(F^{-1}(y))$, output $\mathsf{Dec}_K(C)$

Encrypted message: $F(x), \mathsf{Enc}_{h(x)}(M)$

The trapdoor permutation is only used to communicate a "session key" $h(x)$, the actual message is protected by a symmetric authenticated encryption scheme. The adversary $\mathcal{A}$ in the $\mathsf{PubK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}$ game has no influence over the input of F.

If hash function h is replaced with a "random oracle" (something that just picks a random output value for each input from $\mathcal{X}$), the resulting public-key encryption scheme Π' is CCA secure.

"Textbook" RSA encryption

Key generation

- Choose random prime numbers p and q (each ≈ 1024 bits long)
- $n := pq$ (≈ 2048 bits = key length) $\varphi(n) = (p-1)(q-1)$
- pick integer values e, d such that: $ed \bmod \varphi(n) = 1$
- public key $PK := (n, e)$
- secret key $SK := (n, d)$

Encryption

- input plaintext $M \in \mathbb{Z}_n^*$, public key (n, e)
- $C := M^e \bmod n$

Decryption

- input ciphertext $C \in \mathbb{Z}_n^*$, secret key (n, d)
- $M := C^d \bmod n$

In $\mathbb{Z}_n$: $(M^e)^d = M^{ed} = M^{ed \bmod \varphi(n)} = M^1 = M$.

Common implementation tricks to speed up computation:

- Choose small e with low Hamming weight (e.g., 3, 17, $2^{16}+1$) for faster modular encryption
- Preserve factors of n in $SK = (p, q, d)$, decryption in both $\mathbb{Z}_p$ and $\mathbb{Z}_q$, use Chinese remainder theorem to recover result in $\mathbb{Z}_n$.

“Textbook” RSA is not secure

There are significant security problems with a naive application of the basic “textbook” RSA encryption function $C := P^e \bmod n$:

- deterministic encryption: cannot be CPA secure
- malleability:
  - adversary intercepts C and replaces it with $C' := X^e \cdot C$
  - recipient decrypts $M' = \mathrm{Dec}_{SK}(C') = X \cdot M \bmod n$
- chosen-ciphertext attack recovers plaintext:
  - adversary intercepts C and replaces it with $C' := R^e \cdot C \bmod n$
  - decryption oracle provides $M' = \mathrm{Dec}_{SK}(C') = R \cdot M \bmod n$
  - adversary recovers $M = M' \cdot R^{-1} \bmod n$
- Small value of M (e.g., 128-bit AES key), small exponent $e = 3$:
  - if $M^e < n$ then $C = M^e \bmod n = M^e$ and then $M = \sqrt[3]{C}$ can be calculated efficiently in $\mathbb{Z}$ (no modular arithmetic!)
- many other attacks exist . . .

Using RSA as a CCA-secure encryption scheme

Solution 1: use only as trapdoor function to build encryption scheme

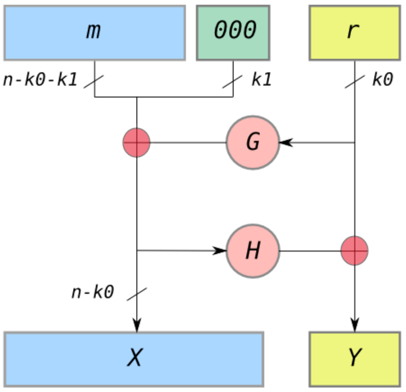
- Pick random value $x \in \mathbb{Z}_n^*$
- Ciphertext is $(x^e \bmod n, \mathrm{Enc}_{h(x)}(M))$, where Enc is from an authenticated encryption scheme

Solution 2: Optimal Asymmetric Encryption Padding

Make M (with zero padding) the left half, and a random string R the right half, of the input of a two-round Feistel cipher, using a secure hash function as the round function.

Interpret the result (X, Y) as an integer M'.

Then calculate $C := M'^e \bmod n$.



Wikipedia/Ozga

PKCS #1 v2.0

Practical pitfalls with implementing RSA

- low entropy of random-number generator seed when generating p and q (e.g. in embedded devices):
  - take public RSA modulus n_1 and n_2 from two devices
  - test $\gcd(n_1, n_2) \overset{?}{=} 1 \Rightarrow$ if no, n_1 and n_2 share this number as a common factor
  - February 2012 experiments: worked for many public HTTPS keys

    Lenstra et al.: Public keys, CRYPTO 2012
    Heninger et al.: Mining your Ps and Qs, USENIX Security 2012.

Outlook

Goals of this course were

- revisit some of the constructions discussed in Part IB security, with emphasis on concrete definitions of security
- introduce some of the discrete algebra necessary to understand public-key encryption schemes, using RSA as an example

Modern cryptography is still a young discipline (born in the early 1980s), but well on its way from a collection of tricks to a discipline with solid theoretical foundations.

Some important concepts that we did not cover here:

- elliptic-curve groups
- digital signatures
- identity-based encryption
- side-channel attacks
- application protocols: electronic voting, digital cash, etc.
- secure multi-party computation