

scalable	fixed-size
public key	secret key
theoretical	practical

Figure: Three dichotomies in cryptography

Three conditions are of central importance:

- ▶ correctness: we must have $x^* = x$.
- ▶ efficiency: the functions enc and dec should be *easy* to execute.
- ▶ security: just using the publicly available information like y (and maybe other items, depending on the system) it should be *hard* to compute x .

We consider a cryptographic transmission protocol as above, with $\text{enc}_{\text{pk}}(x) = y$ and $\text{dec}_{\text{sk}}(y) = x$. When can we consider it secure? Or rather: when is it insecure? We list in decreasing order of success some things that an attacker might achieve, knowing y and the public information (which includes pk in the case of an asymmetric system):

- ▶ for any x , compute sk ,
- ▶ for any x , compute x ,
- ▶ for most x , compute x ,
- ▶ for some x , compute x ,
- ▶ for some x , find out something nontrivial about x ,
- ▶ for some x , possibly find out something nontrivial about x .
- ▶ distinguish statistically between encryptions and random values.

Definition

Let $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$ provide a mapping $\{0, 1\}^n \rightarrow \{0, 1\}^{k(n)}$ for each $n \in \mathbb{N}$, with a polynomially bounded function k of n . An *inversion algorithm* \mathcal{A} for f with *success probability* $\varepsilon(n)$ is such that for all $n \in \mathbb{N}$,

$$\text{prob}(f(\mathcal{A}(z)) = z) = \varepsilon(n),$$

where $z \xleftarrow{\$} f(\{0, 1\}^n)$ is chosen uniformly at random in the image of $\{0, 1\}^n$ under f . Then f is a *one-way function* if

- (i) f can be computed in random polynomial time,
- (ii) all polynomial-time inversion algorithms for f have negligible success probability.
- (iii) Furthermore, f is a *trapdoor function* if there is some secret key sk_n for each n , of polynomially bounded length, so that f has a random polynomial time inversion algorithm \mathcal{A} which takes both $z \xleftarrow{\$} f(\mathbb{B}^n)$ and sk_n as inputs and has nonnegligible success probability.

The product of two $n/2$ -bit integers, possibly with leading zeros, is an n -bit integer, again allowing leading zeros. Thus we get the multiplication function

$f_n: \{0, 1\}^n = \{0, 1\}^{n/2} \times \{0, 1\}^{n/2} \longrightarrow \{0, 1\}^n$. It is easy to compute. Inverting it means factoring the product number z into two $n/2$ -bit numbers. More closely related to RSA is the function f'_n which returns the product if both inputs are prime numbers in the appropriate range, and 0 otherwise. It is conjectured that factoring such products is hard; then f' is a one-way function.

1. *Existential forgery*: The attacker produces a signature s of some message m of his choice.
2. *Selective forgery*: The attacker chooses a message m , then gets access to the public key used for verification, and produces a signature s of m .
3. *Universal forgery*: The attacker receives a message m and produces a signature s of m .
4. *Key recovery*: Given the public key for verification, the attacker produces the secret key for signing.

These four goals are sorted by difficulty:

existential forgery \leq selective forgery
 \leq universal forgery \leq key recovery.

1. *Key only*: The attacker knows the public key, as everybody does.
2. *Known message*: There is a set M of messages, and a signature of each $m \in M$ is known.
3. *Nonadaptively chosen message*: The attacker chooses and submits a set M of messages and receives a signature for each $m \in M$.
4. *Chosen message*: The attacker submits during his computation messages m and receives a signature of each of them.

These resources are ordered in a natural way:

chosen message \leq nonadaptively chosen messages
 \leq known message \leq key only.

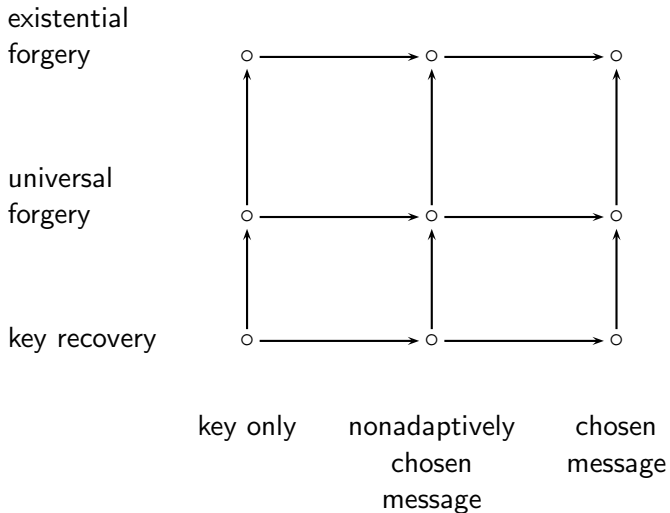


Figure: Relations between attack goals and resources.

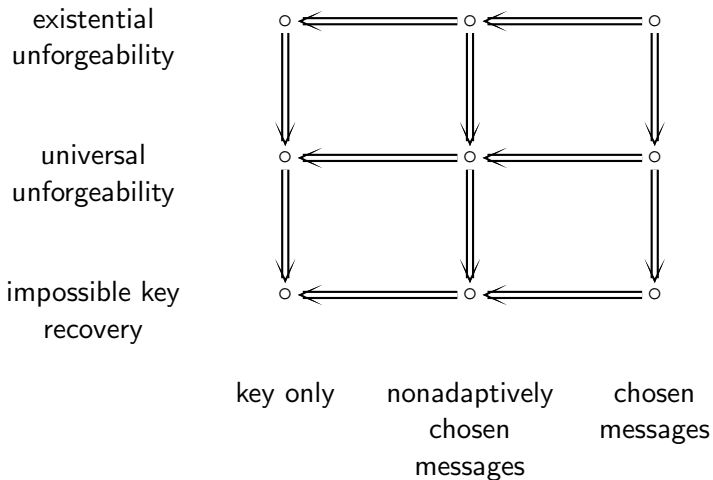


Figure: Implications between security notions.

Theorem

Let A, B with inputs $(x, y), (x, z)$, respectively, be problems as above, and \mathcal{A} a worst-case (with respect to x) reduction from A to B , with success probability σ and time τ . Furthermore, we have a distribution \mathcal{D} for the common inputs x . Then \mathcal{A} also works as an average-case reduction from the distributional versions of A and B , with $x \stackrel{\mathcal{D}}{\leftarrow}$ and the same parameters σ and τ .

PROBLEM SCHEME 1. Distributional RSA problem.

Given: A distribution \mathcal{D} on RSA moduli.

Input: An RSA modulus $N \xleftarrow{\mathcal{D}} \mathcal{D}$ and $y \in \mathbb{Z}_N^\times$.

Output: $x \in \mathbb{Z}_N^\times$ and an integer $c \geq 2$ with $x^c = y$ in \mathbb{Z}_N^\times .

Again we have a trivial reduction

distributional weak RSA problem \leq weak RSA problem.

Assumption

RSA assumption. There is no probabilistic polynomial-time algorithm that computes x from N , e and y in the RSA notation, with nonnegligible success probability.

Strong RSA assumption for GHR; SRSA

For ℓ and n as in the GHR signature scheme and $\mathcal{D} = \mathcal{D}_{n,\ell}$, the distributional weak RSA problem is hard to solve.

Corollary

If the Strong RSA assumption holds, then GHR signatures are existentially unforgeable with chosen messages attacks.

A *key encapsulation mechanism* (KEM) M consists of three probabilistic polynomial-time algorithms $M = (\text{gen}, \text{encap}, \text{decap})$, as follows. We have finite sets K for session keys and C for capsules.

$$\text{gen}(1^n) = (\text{pk}, \text{sk}),$$

$$\text{encap}_{\text{pk}} = (k, c) \in K \times C,$$

$$\text{decap}_{\text{sk}}(c) = k^*.$$

ENCRYPTION SCHEME FROM KEM M 2.

Encryption.

Input: $x \in K$.

Output: $\text{enc}_{\text{pk}}(x) \in C^2$.

1. $(k, c) \xleftarrow{\text{encap}_{\text{pk}}}$
2. $c^* \leftarrow x \cdot k$.
3. Return (c^*, c) .

Decryption.

Input: $(c_1, c_2) \in C^2$.

Output: $\text{dec}_{\text{sk}}(c_1, c_2) \in K$.

4. $k^* \leftarrow \text{decap}_{\text{sk}}(c_2), x^* \leftarrow c_1/k$.
5. Return x^* .

SCHEME 3. Hofheinz-Kiltz key encapsulation mechanism.

Algorithm key generation gen.

Input: 1^n for some integer $n \geq 3$, and $m < n$.

Output: Secret and public keys sk and pk, and a hash function h .

1. $N \xleftarrow{\text{RNG}} \text{sgen}(1^n)$ [Then $N = pq$ is an n -bit integer, and p and q are safe primes of roughly equal size.]
2. $g \xleftarrow{\text{RNG}} \square_N$.
3. $\alpha \xleftarrow{\text{RNG}} \{1, \dots, (N-1)/4\}$.
4. $x \leftarrow g^{\alpha 2^{\ell+m}}$.
5. Return $\text{sk} = (N, g, \alpha)$, $\text{pk} = (N, g, x)$, and a second-preimage resistant hash function $h: \mathbb{Z}_N \rightarrow \mathbb{Z}_{2^m}$.

SCHEME 4. Hofheinz-Kiltz key encapsulation mechanism.

Algorithm Encapsulation encap .

Input: pk .

Output: encap_{pk} .

1. $r \xleftarrow{\$} \{1, \dots, (N-1)/4\}$,
 $R \leftarrow g^{r2^{\ell+m}} \in \square_N$,
 $t = h(R) \in \{1, \dots, 2^\ell - 1\}$,
 $S = |(g^t x)^r| \in \mathbb{Z}_N^\times \cap \{1, \dots, (N-1)/2\}$,
 $u = g^{r2^m}$.
2. $k \leftarrow \text{BBS}_N(u) \in \{0, 1\}^\ell$, $C \leftarrow (R, S)$.
3. Return $\text{encap}_{\text{pk}} = (k, C)$.

SCHEME 5. Hofheinz-Kiltz key encapsulation mechanism.

Algorithm Decapsulation decap.

Input: sk and capsule $(R, S) \in (\mathbb{Z}_N)^2$.

Output: $k^* \in \{0, 1\}^\ell$.

1. If not $R \in \mathbb{Z}_N^\times$ or not $S \in \mathbb{Z}_N^\times \cap \{1, \dots, (N-1)/2\}$ then return “failure”.
2. $t \leftarrow h(R)$.
3. If not

$$(S^2)^{2^{\ell+m}} = (R^2)^{t+\alpha 2^{\ell+m}} \quad (1)$$

then return “failure”.

4. Compute $a, b, c \in \mathbb{Z}$ with $|b| \leq 2^m$, $0 \leq c < m$, and

$$2^c = \gcd(t, 2^{\ell+m}) = at + b2^{\ell+m},$$

5. $T \leftarrow (S^a R^{b-a\alpha})^{2^{m-c}}$,
 $k^* \leftarrow \text{BBS}_N(T) \in \{0, 1\}^\ell$.
6. Return k^* .

Theorem

The Hofheinz-Kiltz key encapsulation Scheme 5 is correct and efficient. That is, in the notation of the scheme we have $k = k^*$, and all operations can be performed probabilistically in time polynomial in n .

Theorem

Let \mathcal{A} be a distinguisher for the KEM Scheme 5 using chosen ciphertexts with success probability at least $\sigma_{\mathcal{A}}$. Then the reduction \mathcal{D} below distinguishes BBS with probability $\sigma_{\mathcal{D}}$, or factors N with probability σ_f , or finds second preimages for h with probability σ_s . Their running times and success probabilities satisfy

$$\tau_{\mathcal{D}}, \tau_{\mathcal{B}} \approx \tau_{\mathcal{A}}, \quad \sigma_{\mathcal{D}} + \sigma_f + \sigma_s \geq \sigma_{\mathcal{A}} - 2^{-n+3}.$$

ALGORITHM 6. BBS distinguisher \mathcal{D} .

Input: (N, z, v) with $z \in \square_N$ and $v \in \mathbb{B}^\ell$.

Output: A bit, or T with $v = \text{BBS}_N(T)$, or "failure".

1. $g \leftarrow \square_N$,
 $\beta \leftarrow \{1, \dots, (N-1)/4\}$.
2. $R^* \leftarrow z$,
 $S^* \leftarrow |(R^*)^\beta|$,
 $t^* \leftarrow h(R^*) \in \mathbb{Z}_{2^m}$,
 $x \leftarrow g^{\beta 2^{\ell+m} - t^*}$.
3. Publish $\text{pk} = (N, g, x)$.
4. Submit the capsule $C^* = (R^*, S^*)$ to \mathcal{A} .
5. When \mathcal{A} submits a decapsulation request for $C = (R, S)$, do steps (6).
6. $t \leftarrow h(R)$.
7. If $R \in \mathbb{Z}_N^\times$, $S \in \mathbb{Z}_N^\times \cap \{1, \dots, (N-1)/2\}$, and

$$(S^2)^{2^{\ell+m}} = (R^2)^{t-t^*+\beta 2^{\ell+m}} \quad (2)$$

then go to step (9) else return "failure". [If the condition holds, we call C *consistent*.]

8. If $t \neq t^*$ then do steps (13).
9. Compute $a^*, b^*, c^* \in \mathbb{Z}$ with $|b^*| < 2^{\ell+m}$, $c^* < m$, and

$$2^{c^*} = \text{gcd}(t - t^*, 2^{\ell+m}) = a^*(t - t^*) + b^* 2^{\ell+m}.$$

10. $T^* \leftarrow (S^{a^*} R^{b^* - a^* \beta})^{2^{m-c^*}}$.
11. Deliver $k = \text{BBS}_N(T^*)$ to \mathcal{A} .
12. If $t = t^*$ then do steps (13).
13. If $R = R^*$ then return $\text{gcd}(S - S^*, N)$. [This is a proper factor of N .]
14. If $R \neq R^*$, then return the collision (R, R^*) of h .
15. When \mathcal{A} returns the decapsulation k^* of C^* , then return 1 if $k^* = v$ and 0 otherwise.

There are three winning situations for \mathcal{D} :

- ▶ Step (13): factor N .
- ▶ Step (14): find second preimage for h .
- ▶ Step (15): distinguish a BBS-value v from random v .

It is sufficient to show the following claims.

1. The above statements about the three winning situations are correct.
2. The probability for at least one of them to happen is large.
3. \mathcal{A} 's decapsulation requests are served properly.
4. \mathcal{D} works efficiently.

Description

- ▶ *Malleability*: The attacker is given a challenge ciphertext and transforms it into another one so that the corresponding plaintexts are distinct but meaningfully related.
- ▶ *Distinguishing*: The attacker presents a plaintext and a second one (of equal length) is chosen uniformly at random from the message space. Then one of the two is chosen uniformly randomly and its encryption is given as the challenge ciphertext to the attacker, whose task it is to determine which one was chosen.
- ▶ *Deciphering*: The attacker is given a challenge ciphertext and finds the corresponding plaintext.
- ▶ *Key recovery*: The attacker computes the secret key.

These four goals are sorted by difficulty:

malleability \leq distinguishing \leq deciphering \leq key recovery.

Definition

1. The *distinguishing power* $\Delta_{\mathcal{D}}$ of \mathcal{D} between X and Y is

$$\Delta_{\mathcal{D}}(X, Y) = |E(\mathcal{D}(X)) - E(\mathcal{D}(Y))|.$$

If $\Delta_{\mathcal{D}}(X, Y) \geq \epsilon$ and \mathcal{D} takes time at most τ , it is a (τ, ϵ) -*distinguisher* between X and Y .

2. X and Y are (τ, ϵ) -*indistinguishable* if

$$\sigma_{\mathcal{D}}(X, Y) \leq \epsilon$$

for all distinguishers \mathcal{D} that take time at most τ .

3. X and Y are *computationally indistinguishable* if $\sigma_{\mathcal{D}}(X, Y)$ is negligible for all polynomial-time distinguishers \mathcal{D} .

If the difference inside the absolute value for $\Delta_{\mathcal{D}}(X, y)$ is negative, then we may switch the outputs 0 and 1 of \mathcal{D} and find for this new distinguisher \mathcal{D}

$$\Delta_{\mathcal{D}}(X) = E(\mathcal{D}(X)) - E(\mathcal{D}(Y)). \quad (3)$$

Now we specify the work of a distinguisher \mathcal{D} between encryptions in more detail.

1. \mathcal{D} generates a plaintext x_0 .
2. \mathcal{D} receives a uniformly random plaintext x_1 and a challenge ciphertext y . [$y = \text{enc}_{\text{pk}}(x_i)$ for a uniformly random $i \in \{0, 1\}$, which is unknown to \mathcal{D} .]
3. \mathcal{D} outputs $i^* \in \{0, 1\}$.

\mathcal{D} is successful if and only if $i = i^*$.

What is the success probability $\sigma_{\mathcal{D}}$ of \mathcal{D} as above? We use $\Delta_{\mathcal{D}}(X)$, for the distributions X_0 and X_1 of $\text{enc}_{\text{pk}}(x_0)$ and $\text{enc}_{\text{pk}}(x_1)$, respectively. Then

$$\begin{aligned}\sigma_{\mathcal{D}} &= \text{prob}\{\mathcal{D}(y) = 0 : y \leftarrow X_0\} \cdot \text{prob}\{i = 0\} \\ &\quad + \text{prob}\{\mathcal{D}(y) = 1 : y \leftarrow X_1\} \cdot \text{prob}\{i = 1\} \\ &= (1 - E(\mathcal{D}(X_0))) \cdot \frac{1}{2} + E(\mathcal{D}(X_1)) \cdot \frac{1}{2} \\ &= \frac{1}{2}(1 + \Delta_{\mathcal{D}}(X_0, X_1)).\end{aligned}\tag{4}$$

Description

- ▶ *Key only*: The attacker knows the public key, as everybody does.
- ▶ *Non-adaptively chosen ciphertexts*: The attacker may submit ciphertexts of his choice to a decryption oracle, and after this phase receives the challenge ciphertext.
- ▶ *(Adaptively) chosen ciphertexts*: The attacker submits ciphertexts to the decryption oracle, then receives the challenge ciphertext, and then may submit further ciphertexts (distinct from the challenge) to a decryption oracle. We usually leave out the “adaptively”.

These resources are ordered in a natural way:

key only \leq non-adaptively chosen ciphertexts
 \leq chosen ciphertexts.

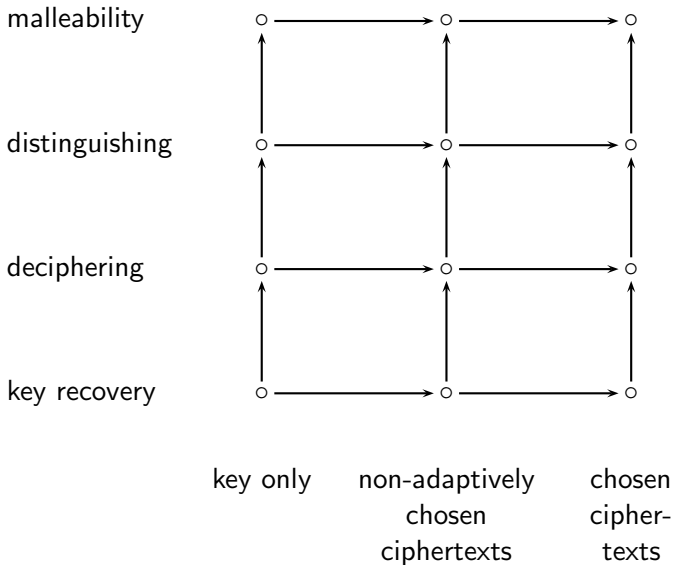


Figure: Relations between attack goals and resources.

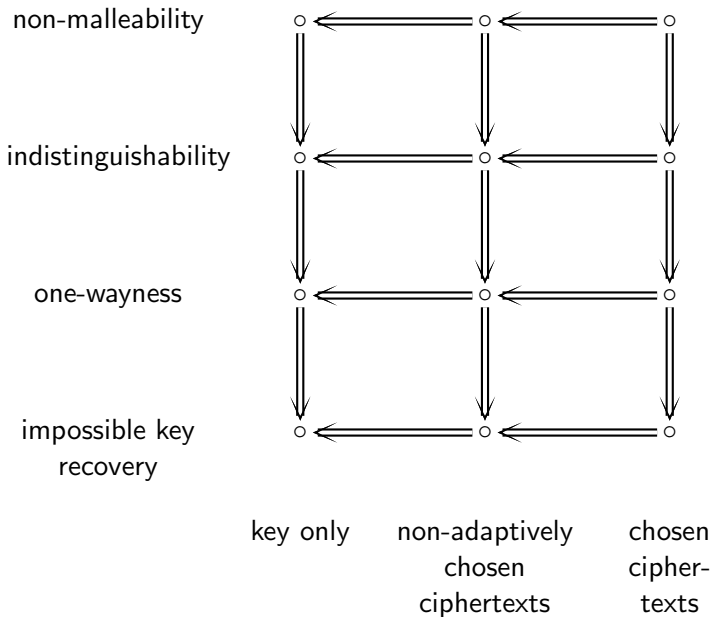


Figure: Implications between security notions for encryption.

CRYPTOSYSTEM 7. ElGamal encryption scheme.

General setup:

Input: 1^n .

1. Choose a finite cyclic group $G = \langle g \rangle$ with d elements, where d is an n -bit integer, and a generating element g . These data, including a description of G , are made public.

Key generation:

Output: (sk, pk) .

2. Choose a secret key $sk = b \xleftarrow{\$} \mathbb{Z}_d$ at random. The public key is $pk = B = g^b \in G$, and $sk = b$ is the secret key.

CRYPTOSYSTEM 8. ElGamal encryption scheme.

Encryption:

Input: Plaintext $x \in G$.

Output: Ciphertext $\text{enc}_{\text{pk}}(x) \in G^2$.

1. Choose a secret session key $a \xleftarrow{\$} \mathbb{Z}_d$ at random.
2. Public session key $A \xleftarrow{\$} g^a \in G$, and common session key $k \leftarrow B^a = g^{ab} = A^b$.
3. $y \leftarrow x \cdot k \in G$, Return $\text{enc}_{\text{pk}}(x) = (y, A)$.

Decryption:

Input: Arbitrary $(y, A) \in G^2$.

Output: Decryption $\text{dec}_{\text{sk}}(y, A) \in G$.

4. Common session key $k \leftarrow A^b$, inverse $k^{-1} \in G$ of the common key.
5. $z \leftarrow y \cdot k^{-1}$, Return $\text{dec}_{\text{sk}}(y, A) = z$.

REDUCTION \mathcal{R} FROM DDH_G TO AN ELGAMAL DISTINGUISHER \mathcal{D} 9.

Input: $A, B, C \in G$.

Output: 0 or 1.

1. $\text{pk} \leftarrow (d, g, B, \text{description of } G)$.
2. \mathcal{D} chooses a plaintext $x_0 \in G$ and gives it to \mathcal{R} .
3. $x_1 \xleftarrow{\$} G$.
4. $i \xleftarrow{\$} \{0, 1\}$.
5. $e \leftarrow (x_i \cdot C, A)$.
6. \mathcal{R} sends e to \mathcal{D} and receives i^* .
7. If $i = i^*$ then return 1 else return 0.

Theorem

Let $\tau_{\mathcal{D}}$ and $\Delta_{\mathcal{D}}$ denote the running time and distinguishing power, respectively, of \mathcal{D} . Then \mathcal{R} takes time $\tau_{\mathcal{D}} + O(\text{mult}_G)$ and distinguishes DH triples from random triples in G^3 with distinguishing power at least $\Delta_{\mathcal{D}}/2$.

Corollary

If DDH_G is hard, then ElGamal encryption in G is indistinguishable by key only attacks .

$DDH \leq DH \leq DL,$
 $DDH \text{ hard} \implies DH \text{ hard} \implies DL \text{ hard}.$

non-malleable	-	-
indistinguishable	DDH	-
undecipherable	DH	-
impossible key recovery	DL	GDH
<hr/>		
	key only	chosen ciphertexts

Figure: The security profile of ElGamal encryption.