Formal and Strong Security Definitions:
IND-CPA security

There are three kinds of lies:
small lies, big lies and statistics.

Sven Laur
swen@math.ut.ee

Helsinki University of Technology

Basic theoretical notions

Formal syntax of a cryptosystem I

Various domains associated with the cryptosystem:

$\mathcal{M}$ – a set of plausible messages (plaintexts);

$\mathcal{C}$ – a set of possible cryptograms (ciphertexts);

$\mathcal{R}$ – random coins used by the encryption algorithm.

Parameters used by the encryption and decryption algorithms:

pk – a public key (public knowledge needed to generate valid encryptions);

sk – a secret key (knowledge that allows efficient decryption of ciphertexts).

Formal syntax of a cryptosystem II

Algorithms that define a cryptosystem:

$\mathcal{G}$ – a randomised key generation algorithm;

$\mathcal{E}_{\mathsf{pk}}$ – a randomised encryption algorithm;

$\mathcal{D}_{\mathsf{sk}}$ – a deterministic decryption algorithm.

The key generation algorithm $\mathcal{G}$ outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.

The encryption algorithm is an efficient mapping $\mathcal{E}_{\mathsf{pk}} : \mathcal{M} \times \mathcal{R} \rightarrow \mathcal{C}$.

The decryption algorithm is an efficient mapping $\mathcal{D}_{\mathsf{sk}} : \mathcal{C} \rightarrow \mathcal{M}$.

A cryptosystem must be functional

$$\forall (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G},\ \forall m \in \mathcal{M},\ \forall r \in \mathcal{R}: \quad \mathcal{D}_{\mathsf{sk}}(\mathcal{E}_{\mathsf{pk}}(m; r)) = m.$$

Example. RSA-1024 cryptosystem

Key generation $\mathcal{G}$:

1. Choose uniformly 512-bit prime numbers p and q.
2. Compute $N = p \cdot q$ and $\phi(N) = (p-1)(q-1)$.
3. Choose uniformly $e \leftarrow \mathbb{Z}^*_{\phi(N)}$ and set $d = e^{-1} \mod \phi(N)$.
4. Output $\mathsf{sk} = (p, q, e, d)$ and $\mathsf{pk} = (N, e)$.

Encryption and decryption:

$$\mathcal{M} = \mathbb{Z}_N, \quad \mathcal{C} = \mathbb{Z}_N, \quad \mathcal{R} = \emptyset$$

$$\mathcal{E}_{\mathsf{pk}}(m) = m^e \mod N \qquad \mathcal{D}_{\mathsf{sk}}(c) = c^d \mod N \ .$$

When is a cryptosystem secure?

It is rather hard to tell when a cryptosystem is secure. Instead people often specify when a cryptosystem is broken.

- **Complete key recovery:**
  Given pk and $\mathcal{E}_{\mathsf{pk}}(m_1), \ldots, \mathcal{E}_{\mathsf{pk}}(m_n)$, the adversary deduces sk in a *feasible* time with a *reasonable* probability.
- **Complete plaintext recovery:**
  Given pk and $\mathcal{E}_{\mathsf{pk}}(m_1), \ldots, \mathcal{E}_{\mathsf{pk}}(m_n)$, the adversary is able to recover m_i in a *feasible* time with a *reasonable* probability.
- **Partial plaintext recovery:**
  Given pk and $\mathcal{E}_{\mathsf{pk}}(m_1), \ldots, \mathcal{E}_{\mathsf{pk}}(m_n)$, the adversary is able to recover a part of m_i in a *feasible* time with a *reasonable* probability.

Formal approach. Hypothesis testing

We can formalise partial recovery using hypothesis testing:

1. Challenger generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}$.
2. Challenger chooses a message m from a distribution $\mathcal{M}_0$.
3. Challenger sends $c \leftarrow \mathcal{E}_{\mathsf{pk}}(m)$ and pk to Malice.
4. Malice must decide whether a hypothesis $\mathcal{H}$ holds for m or not.

The distribution $\mathcal{M}_0$ characterises Malice's knowledge about the input.

The hypothesis $\mathcal{H}$ can describe various properties of m such as:

- The message m is form a message space $\mathcal{M}_0$ (trivial hypothesis).
- The message m is equal to 0 (simple hypothesis).
- The message m is larger than 500 (complex hypothesis).

Simplest guessing game

Consider the simplest attack scenario:

1. $\mathcal{M}_0$ is a uniform distribution over the messages m_0 and m_1.
2. $\mathcal{H}_0$ and $\mathcal{H}_1$ denote simple hypotheses $[m = m_0]$ and $[m = m_1]$.
3. Malice must choose between these hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$.

The probability of an incorrect guess

$$
\begin{aligned}
\Pr[\textsf{Failure}] &= \Pr[\mathcal{H}_0] \cdot \Pr[\textsf{Malice} = 1|\mathcal{H}_0] + \Pr[\mathcal{H}_1] \cdot \Pr[\textsf{Malice} = 0|\mathcal{H}_1] \\
&= \frac{1}{2} \cdot \bigl(\underbrace{\Pr[\textsf{Malice} = 1|\mathcal{H}_0]}_{\textsf{False negatives}} + \underbrace{\Pr[\textsf{Malice} = 0|\mathcal{H}_1]}_{\textsf{False positives}}\bigr) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \underbrace{\bigl(\Pr[\textsf{Malice} = 1|\mathcal{H}_0] - \Pr[\textsf{Malice} = 1|\mathcal{H}_1]\bigr)}_{\pm\textsf{Adv}(\textsf{Malice})} \quad .
\end{aligned}
$$

IND-CPA security

Malice is good in breaking security of a cryptosystem $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ if Malice can distinguish two games (interactive hypothesis testing):

Game $\mathcal{G}_0$	Game $\mathcal{G}_1$
1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}$ 2. $(m_0, m_1, \sigma) \leftarrow \mathsf{Malice}(\mathsf{pk})$ 3. $\mathsf{guess} \leftarrow \mathsf{Malice}(\sigma, \mathcal{E}_{\mathsf{pk}}(m_0))$	1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}$ 2. $(m_0, m_1, \sigma) \leftarrow \mathsf{Malice}(\mathsf{pk})$ 3. $\mathsf{guess} \leftarrow \mathsf{Malice}(\sigma, \mathcal{E}_{\mathsf{pk}}(m_1))$

with a *non-negligible* advantage*

$$\begin{aligned}\mathsf{Adv}(\mathsf{Malice}) &= \left|\Pr\left[\mathsf{guess} = 0 | \mathcal{G}_0\right] - \Pr\left[\mathsf{guess} = 0 | \mathcal{G}_1\right]\right| \\ &= \left|\Pr\left[\mathsf{guess} = 1 | \mathcal{G}_0\right] - \Pr\left[\mathsf{guess} = 1 | \mathcal{G}_1\right]\right|\end{aligned}$$

*Twice larger than defined in the Mao's book

Is the RSA cryptosystem IND-CPA secure?

What does it mean in practise?

Bit-guessing game with a fair coin

Consider Protocol 14.1 in Mao's book:

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{G}$
2. $(m_0, m_1, \sigma) \leftarrow$ Malice(pk) where σ denotes the internal state.
3. The oracle $\mathcal{O}$ flips a fair coin $b \leftarrow \{0, 1\}$ and sets $c \leftarrow \mathcal{E}_{\mathsf{pk}}(m_b)$.
4. guess $\leftarrow$ Malice(σ, c)

The success probability

$$\begin{aligned}
\Pr[\mathsf{Success}] &= \Pr[b = 0 \wedge \mathsf{guess} = 0] + \Pr[b = 1 \wedge \mathsf{guess} = 1] \\
&= \frac{1}{2} \cdot \Pr[\mathsf{guess} = 0 | \mathcal{G}_0] + \frac{1}{2} \cdot \big(1 - \Pr[\mathsf{guess} = 0 | \mathcal{G}_1]\big) \\
&= \frac{1}{2} \pm \frac{1}{2} \cdot \mathsf{Adv}(\mathsf{Malice})
\end{aligned}$$

Bit-guessing game with a biased coin

For clarity let $\Pr[b = 0] \leq \Pr[b = 1]$. Then

$$\begin{aligned}\Pr[\mathsf{Success}] &\leq \Pr[b = 1] \cdot \big(\Pr[\mathsf{guess} = 0|\mathcal{G}_0] + \Pr[\mathsf{guess} = 1|\mathcal{G}_1]\big)\\ &\leq \Pr[b = 1] + \Pr[b = 1] \cdot \mathsf{Adv}(\mathsf{Malice}) \enspace ,\end{aligned}$$

$$\begin{aligned}\Pr[\mathsf{Success}] &\geq \Pr[b = 0] \cdot \big(\Pr[\mathsf{guess} = 0|\mathcal{G}_0] + \Pr[\mathsf{guess} = 1|\mathcal{G}_1]\big)\\ &\geq \Pr[b = 0] - \Pr[b = 0] \cdot \mathsf{Adv}(\mathsf{Malice}) \enspace .\end{aligned}$$

Hence, the advantage determines guessing precision

$$\Pr[b = 0] - \mathsf{Adv}(\mathsf{Malice}) \leq \Pr[\mathsf{Success}] \leq \Pr[b = 1] + \mathsf{Adv}(\mathsf{Malice}) \enspace .$$

Beyond bit-guessing games

The coin-flipping game is a simplified setting, where the input distribution $\mathcal{M}_0$ is defined over $\{m_0, m_1\}$ and Malice must choose between m_0 and m_1.

But there are more general cases:

- $\mathcal{M}_0$ might be defined over many elements of $\mathcal{M}$.
- Malice might accept or reject complex hypotheses $\mathcal{H}$.
- Malice might try to test many hypotheses $\mathcal{H}_1, \ldots, \mathcal{H}_s$ simultaneously.
- Malice might try to predict a function $g(m)$.

All these settings can be modelled as prediction tasks, where Malice specifies the input distribution $\mathcal{M}_0$. What are the corresponding functions?

Semantic security

Consider a complex attack scenario:

1. The oracle $\mathcal{O}$ runs $\mathcal{G}$ and sends pk to Charlie.
2. Charlie describes a distribution $\mathcal{M}_0$ to the oracle $\mathcal{O}$.
3. The oracle $\mathcal{O}$ samples $m \leftarrow \mathcal{M}_0$ and sends $c \leftarrow \mathcal{E}_{\mathsf{pk}}(m)$ to Charlie.
4. Charlie outputs his guess guess of $g(m)$.

Trivial attack

Always choose a prediction i of $g(m)$ that maximises $\Pr\left[g(m) = i|\mathcal{M}_0\right]$.

Normalised guessing advantage

$$\mathsf{Adv}^{\mathsf{guess}}(\mathsf{Charlie}) = \Pr\left[\mathsf{guess} = g(m)\right] - \underbrace{\max\left\{\Pr\left[g(m) = i|\mathcal{M}_0\right]\right\}}_{\mathsf{Adv}(\mathsf{Triv})}$$

IND-CPA security implies semantic security

If Charlie is good at predicting an efficiently computable function $g : \mathcal{M} \to \mathbb{Z}$ then we can construct an efficient IND-CPA adversary Malice:

1. Malice forwards pk to Charlie.
2. Charlie describes $\mathcal{M}_0$ to Malice.
3. Malice independently samples $m_0 \leftarrow \mathcal{M}_0$ and $m_1 \leftarrow \mathcal{M}_0$.
4. Malice forwards $c = \mathcal{E}_{\mathsf{pk}}(m_b)$ to Charlie.
5. Charlie outputs his guess guess to Malice who
   – outputs 0 if $\mathsf{guess} = g(m_0)$,
   – outputs 1 if $\mathsf{guess} \neq g(m_0)$.

Running time

If $g(m_0)$ is efficiently computable and sampling procedure for the distribution $\mathcal{M}_0$ is efficient then Malice and Charlie have comparable running times.

How well does Malice perform?

In both games Malice outputs 0 only if guess $= g(m_0)$ and thus

$$\Pr\left[\mathsf{Malice} = 0 | \mathcal{G}_0\right] = \mathsf{Adv}^{\mathsf{guess}}(\mathsf{Charlie}) + \mathsf{Adv}(\mathsf{Triv}) \quad ,$$

$$\Pr\left[\mathsf{Malice} = 0 | \mathcal{G}_1\right] = \sum_{\mathsf{pk}, c, r_{ch}} \Pr\left[\mathsf{pk}, c, r_{ch}\right] \cdot \Pr\left[\mathsf{guess} = g(m_0) | \mathsf{pk}, c, r_{ch}, \mathcal{G}_1\right] \quad ,$$

where r_{ch} denotes the random coins used by Charlie. As the triple (pk, c, r_{ch}) completely determines the reply guess, we can express

$$\begin{aligned}\Pr\left[\mathsf{guess} = g(m_0) | \mathsf{pk}, c, r_{ch}, \mathcal{G}_1\right] &= \Pr\left[m_0 \leftarrow \mathcal{M}_0 : g(m_0) = \mathsf{guess}\right] \\ &\leq \max\left\{\Pr\left[g(m) = i | \mathcal{M}_0\right]\right\} = \mathsf{Adv}(\mathsf{Triv}) \quad .\end{aligned}$$

How well does Malice perform?

Thus, we obtain

$$\Pr\left[\mathsf{Malice} = 0 | \mathcal{G}_0\right] = \mathsf{Adv}^{\mathsf{guess}}(\mathsf{Charlie}) + \mathsf{Adv}(\mathsf{Triv}) \ ,$$

$$\begin{aligned} \Pr\left[\mathsf{Malice} = 0 | \mathcal{G}_1\right] &= \sum_{\mathsf{pk}, c, r_{ch}} \Pr\left[\mathsf{pk}, c, r_{ch}\right] \cdot \Pr\left[\mathsf{guess} = g(m_0) | \mathsf{pk}, c, r_{ch}, \mathcal{G}_1\right] \\ &\leq \sum_{\mathsf{pk}, c, r_{ch}} \Pr\left[\mathsf{pk}, c, r_{ch}\right] \cdot \mathsf{Adv}(\mathsf{Triv}) = \mathsf{Adv}(\mathsf{Triv}) \ . \end{aligned}$$

In other words Charlie and Malice have the same advantage

$$\mathsf{Adv}(\mathsf{Malice}) = \left|\Pr\left[\mathsf{Malice} = 0 | \mathcal{G}_0\right] - \Pr\left[\mathsf{Malice} = 0 | \mathcal{G}_1\right]\right| \geq \mathsf{Adv}^{\mathsf{guess}}(\mathsf{Charlie}) \ .$$

What if the function g is not efficiently computable?

What if $\mathcal{M}_0$ cannot be sampled efficiently?

What does it mean in practise?

Historical references

Shaft Goldwasser and Silvio Micali, *Probabilistic Encryption & How To Play Mental Poker Keeping Secret All Partial Information*, 1982.

- Non-adaptive choice of $\mathcal{M}_0$ and semantic security for any function.

Contemporary treatment of semantic security:

- Mihir Bellare, Anand Desai, E. Jokipii and Phillip Rogaway, *A Concrete Security Treatment of Symmetric Encryption*, 1997.
- Mihir Bellare, Anand Desai, David Pointcheval and Phillip Rogaway, *Relations among Notions of Security for Public-Key Encryption Schemes*, 1998.

Mental poker

Commutative cryptosystems

A cryptosystem $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is commutative if for any valid public keys $\mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}$

$$\forall m \in \mathcal{M} : \quad \mathcal{E}_{\mathsf{pk}_\mathsf{A}}(\mathcal{E}_{\mathsf{pk}_\mathsf{B}}(m)) = \mathcal{E}_{\mathsf{pk}_\mathsf{B}}(\mathcal{E}_{\mathsf{pk}_\mathsf{A}}(m)).$$

In particular it implies

$$m = \mathcal{D}_{\mathsf{sk}_\mathsf{A}}(\mathcal{D}_{\mathsf{sk}_\mathsf{B}}(\mathcal{E}_{\mathsf{pk}_\mathsf{A}}(\mathcal{E}_{\mathsf{pk}_\mathsf{B}}(m)))) = \mathcal{D}_{\mathsf{sk}_\mathsf{B}}(\mathcal{D}_{\mathsf{sk}_\mathsf{A}}(\mathcal{E}_{\mathsf{pk}_\mathsf{B}}(\mathcal{E}_{\mathsf{pk}_\mathsf{A}}(m)))).$$

The latter allows to swap the order of encryption and decryption operations.

Mental poker protocol

1. Alice sends randomly shuffled encryptions $\mathcal{E}_{\mathsf{pk_A}}(\spadesuit 2), \dots, \mathcal{E}_{\mathsf{pk_A}}(\heartsuit \mathsf{A})$.
2. Bob chooses randomly c_A, c_B and sends $c_A, \mathcal{E}_{\mathsf{pk_B}}(c_B)$ to Alice.
3. Alice sends $\mathcal{D}_{\mathsf{sk_A}}(\mathcal{E}_{\mathsf{pk_B}}(c_B))$ to Bob and locally outputs $\mathcal{D}_{\mathsf{sk_A}}(c_A)$.
4. Bob outputs locally $\mathcal{D}_{\mathsf{sk_B}}(\mathcal{D}_{\mathsf{sk_A}}(\mathcal{E}_{\mathsf{pk_B}}(c_B))) = \mathcal{D}_{\mathsf{sk_A}}(c_B)$.
5. Alice sends her $\mathsf{pk_A}$ to Bob. Bob sends his $\mathsf{pk_B}$ to Alice.

RSA with shared modulus $N = pq$, and keys $(\mathsf{pk_A}, \mathsf{sk_A}) = (e_A, d_A)$ and $(\mathsf{pk_B}, \mathsf{sk_B}) = (e_B, d_B)$ such that

$$e_A d_A = 1 \mod \phi(N) \qquad e_B d_B = 1 \mod \phi(N)$$

is insecure after Step 5. **Why?**

Attacks against mental poker game

Recall that RSA encryption preserves quadratic residuocity and both parties can compute it. Leaking residuocity can give an edge to Bob.

Brute force attack. Let $\spadesuit 2, \ldots, \heartsuit$A be encoded as $1, \ldots, 52$. Then corresponding encryptions are $1, 2^{e_A}, \ldots, 56^{e_A}$ modulo N. Obviously,

$$2^{e_A} \cdot 2^{e_A} = 4^{e_A} \mod N, \quad \ldots, \quad 7^{e_A} \cdot 7^{e_A} = 49^{e_A} \mod N$$

and Bob can with high probability separate encryptions of $2, \ldots, 7$.

Similar connections allow Bob to reveal most of the cards.

There are completely insecure encodings for the cards:

– Vanilla RSA is not applicable for secure encryption.

– Vanilla RSA is not IND-CPA secure.

IND-CPA secure cryptosystems

Goldwasser-Micali cryptosystem

Famous conjecture. Let N be a large RSA modulus. Then without factorisation of N it is infeasible to determine whether a random $c \in J_N(1)$ is a quadratic residue or not.

Key generation. Generate safe primes $p, q \in \mathbb{P}$ and choose quadratic non-residue $y \in J_N(1)$ modulo $N = pq$. Set $\mathsf{pk} = (N, y)$, $\mathsf{sk} = (p, q)$.

Encryption. First choose a random $x \leftarrow \mathbb{Z}_N^*$ and then compute

$$\mathcal{E}_{\mathsf{pk}}(0) = x^2 \mod N \quad \text{and} \quad \mathcal{E}_{\mathsf{pk}}(1) = yx^2 \mod N.$$

Decryption. Given c, compute $c_1 \mod p$ and $c_2 \mod q$ and use Euler's criterion to test whether c is a quadratic residue or not.

ElGamal cryptosystem

Combine the Diffie-Hellman key exchange protocol

Alice		**Bob**
$x \leftarrow \mathbb{Z}_{\|G\|}$	$\xrightarrow{y=g^x}$	$k \leftarrow \mathbb{Z}_{\|G\|}$
	$\xleftarrow{g^k}$	
$g^{xk} = (g^k)^x$		$g^{xk} = (g^x)^k$

with one-time pad using multiplication in $G = \langle g \rangle$ as encoding rule

$$\mathcal{E}_{\mathsf{pk}}(m) = (g^k, m \cdot g^{xk}) = (g^k, m \cdot y^k) \qquad \text{for all elements } m \in G$$

with a public key $\mathsf{pk} = y = g^x$ and a secret key $\mathsf{sk} = x$.

Decisional Diffie-Hellman Assumption (DDH)

DDH Assumption. For a fixed group G, Charlie can distinguish two games

Game $\mathcal{G}_0$	Game $\mathcal{G}_1$
1. $x, k \leftarrow \mathbb{Z}_q,\ \ q = \lvert G \rvert$ 2. guess $\leftarrow$ Charlie(g, g^x, g^k, g^{xk})	1. $x, k, c \leftarrow \mathbb{Z}_q,\ \ q = \lvert G \rvert$ 2. guess $\leftarrow$ Charlie(g, g^x, g^k, g^c)

with a negligible advantage

$$\mathsf{Adv}(\mathsf{Charlie}) = \left|\Pr\left[\mathsf{guess} = 0 \middle| \mathcal{G}_0\right] - \Pr\left[\mathsf{guess} = 0 \middle| \mathcal{G}_1\right]\right| \quad .$$

The Diffie-Hellman key exchange protocol is secure under the DDH assumption, as Charlie cannot tell the difference between g^{xk} and g^c.

ElGamal is IND-CPA secure

If the Diffie-Hellman key exchange protocol is secure then the ElGamal cryptosystem must be secure, as the one-time pad is unbreakable.

Let Malice be good in IND-CPA game. Now Charlie given (g, g^x, g^k, z):

1. Sets pk $= g^x$ and $(m_0, m_1, \sigma) \leftarrow$ Malice(pk).
2. Tosses a fair coin $b \leftarrow \{0, 1\}$ and set $c = (g^k, m_b z)$.
3. Gets guess $\leftarrow$ Malice(σ, c).
4. If guess $= b$ returns 0 else outputs 1.

We argue that this is a good strategy to win the DDH game:

- In the game $\mathcal{G}_0$, we simulate the bit guessing game.
- In the game $\mathcal{G}_1$, the guess guess is independent form b.

Charlie's advantage in the game $\mathcal{G}_1$

Note that $c = (g^k, m_b z)$ is uniformly chosen from $G \times G$ in the game $\mathcal{G}_1$ and we can rewrite (simplify) the code of Charlie (for the game $\mathcal{G}_1$):

1. Set pk $= g^x$ and $(m_0, m_1, \sigma) \leftarrow$ Malice(pk).
2. Toss a fair coin $b \leftarrow \{0, 1\}$ and set $c = (g^k, c_2)$ for $c_2 \leftarrow G$.
3. Get guess $\leftarrow$ Malice(σ, c).
4. If guess $= b$ return 0 else output 1.

Charlie's advantage in the game $\mathcal{G}_1$

Note that $c = (g^k, m_b z)$ is uniformly chosen from $G \times G$ in the game $\mathcal{G}_1$ and we can rewrite (simplify) the code of Charlie (for the game $\mathcal{G}_1$):

1. Set pk $= g^x$ and $(m_0, m_1, \sigma) \leftarrow$ Malice(pk).
2. Set $c = (g^k, c_2)$ for $c_2 \leftarrow G$.
3. Get guess $\leftarrow$ Malice(σ, c).
4. Toss a fair coin $b \leftarrow \{0, 1\}$. If guess $= b$ return 0 else output 1.

Therefore

$$\Pr\left[\text{Charlie} = 0 | \mathcal{G}_1\right] = \frac{1}{2} \enspace .$$

Charlie's advantage in the DDH game

By combining estimates

$$\begin{aligned}\Pr\left[\mathsf{Charlie} = 0 | \mathcal{G}_1\right] &= \frac{1}{2} \\ \Pr\left[\mathsf{Charlie} = 0 | \mathcal{G}_0\right] &= \Pr\left[\mathsf{Success\ in\ bit\ guessing\ game}\right] \\ &= \frac{1}{2} \pm \frac{1}{2} \cdot \mathsf{Adv}(\mathsf{Malice})\end{aligned}$$

we obtain

$$\mathsf{Adv}(\mathsf{Charlie}) = \frac{1}{2} \cdot \mathsf{Adv}(\mathsf{Malice})$$

Why some instantiations of ElGamal fail?

If the message $m \notin G$ then mg^{xk} is not one-time pad, for example

$$G = \langle 2 \mod 6 \rangle \qquad \Longrightarrow \qquad m2^{xk} = \pm m \mod 3$$

and a single bit of information is always revealed.

Fix a generator of $g \in \mathbb{Z}_p^*$ for large $p \in \mathbb{P}$ such that DDH holds.
If public key $y = g^x$ is quadratic residue (QR), then y^k is also QR.

$$\boxed{m \text{ is QR if and only if } my^k \text{ is QR}}$$

Fix I: Choose $g \in \mathrm{QR}$ so that $\langle g \rangle = \mathrm{QR}$ and $m \in \mathrm{QR}$.

Fix II: Choose almost regular hash function $h : G \to \{0,1\}^\ell$ and define $\mathcal{E}_{\mathsf{pk}}(m) = (g^k, h(g^{xk}) \oplus m)$ for $m \in \{0,1\}^\ell$. Then $h(g^{xk})$ is almost uniform.

Hybrid encryption

Assume that $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a IND-CPA secure cryptosystem and prg is a secure pseudorandom generator (secure stream-cipher, e.g. AES in counter mode).

Encrypt. For $m \in \{0,1\}^{\ell}$ choose seed $\in \mathcal{M}$ randomly and compute

$$\mathcal{E}^*_{\mathsf{pk}}(m) = (\mathcal{E}_{\mathsf{pk}}(\mathsf{seed}), \mathsf{prg}(\mathsf{seed}) \oplus m)$$

Decrypt. Given (c_1, c_2) compute seed $\leftarrow \mathcal{D}_{\mathsf{sk}}(c_1)$ and output $c_2 \oplus \mathsf{prg}(\mathsf{seed})$.

Theorem. The hybrid encryption is IND-CPA secure.

Efficiency considerations

How much time can Malice spend?

Usually, it is assumed that Malice uses a probabilistic polynomial time algorithm to launch the attack. What does it mean?

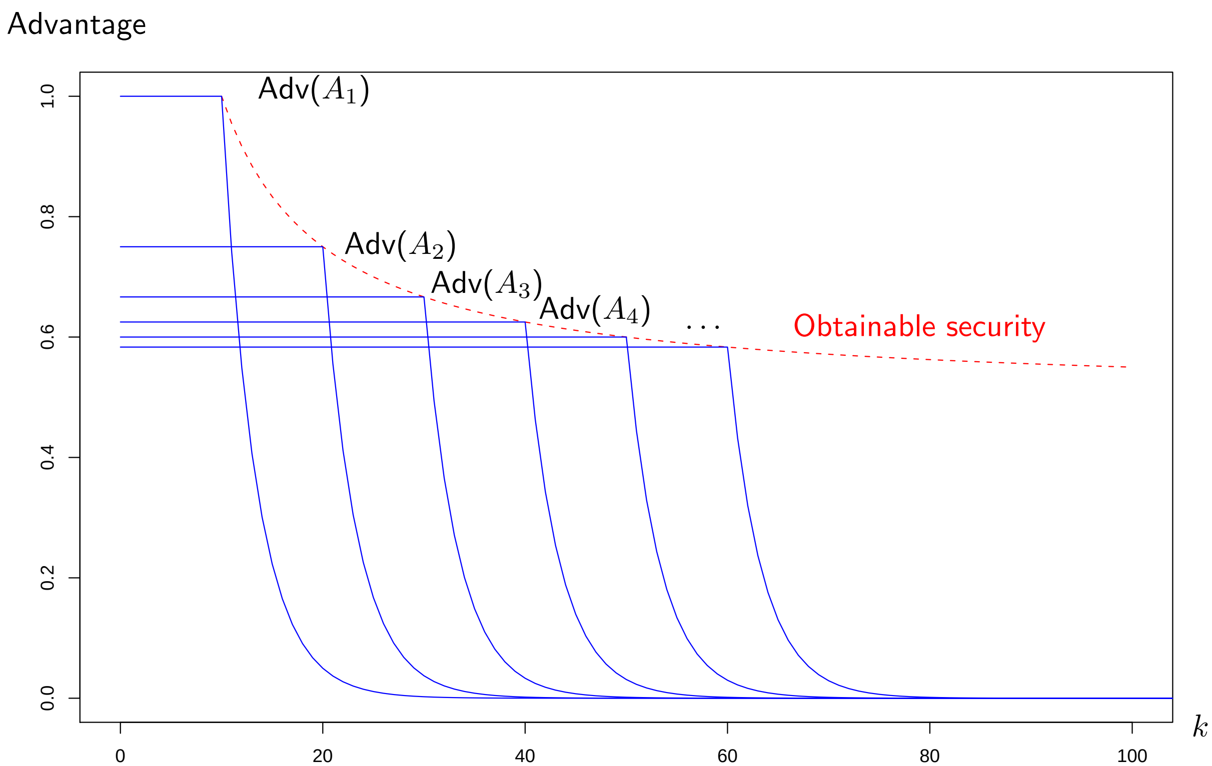
Example
1994 – 426 bit RSA challenge broken.
2003 – 576 bit RSA challenge broken.
2005 – 640 bit RSA challenge broken.

Instead of a concrete encryption scheme RSA is a family of cryptosystems and Malice can run algorithm polynomial in the length k of RSA modulus.

Negligible advantage means that the advantage decreases faster than k^{-c} for any $c > 0$.

A concrete example

For simplicity, imagine that Malice runs algorithms that finish in time k^5.



Uniform *vs* non-uniform security

For each polynomial-time algorithm A_i the advantage was negligible:
$\Longrightarrow$ scheme is secure against polynomial *uniform* adversaries.

If Malice chooses a good algorithm for each k separately
$\Longrightarrow$ she breaks the scheme with advantage $\frac{1}{2}$;
$\Longrightarrow$ scheme is insecure against polynomial *non-uniform* adversaries.

In practise, each adversary has limited resources
$\Longrightarrow$ Given time t, Malice should not achieve $\mathsf{Adv}(\mathsf{Malice}) \geq \varepsilon_{\mathsf{critical}}$.

If scheme is secure against non-uniform adversaries then for large k:
$\Longrightarrow$ $\mathsf{Adv}(\mathsf{Malice}) \leq \varepsilon_{\mathsf{critical}}$ for all t time algorithms;
$\Longrightarrow$ the scheme is still efficiently implementable.

Is non-uniform security model adequate in practise*?

Consider the case of browser certificates:

- Several Verisign certificates have been issued in 1996–1998.
- As a potential adversary knows pk, he can design a special crack algorithm for that pk only. He does not care about other values of pk.
- Maybe a special bit pattern of $N = pq$ allows more efficient factorisation?

Why can't we fix pk in the non-uniform model?

Is there a model that describes reality without problems*?

Does security against (non-)uniform adversaries *heuristically* imply security in real applications*?