# Lecture 7

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

March 6, 2015

# Semantic Security (1/3)

- RSA clearly provides some kind of “security”, but it is clear that we need to be more careful with what we ask for.

# Semantic Security (1/3)

- RSA clearly provides some kind of "security", but it is clear that we need to be more careful with what we ask for.
- Intuitively, we want to leak no information of the encrypted plaintext.

# Semantic Security (1/3)

- RSA clearly provides some kind of “security”, but it is clear that we need to be more careful with what we ask for.
- Intuitively, we want to leak no **knowledge** of the encrypted plaintext.

# Semantic Security (1/3)

- RSA clearly provides some kind of "security", but it is clear that we need to be more careful with what we ask for.
- Intuitively, we want to leak no **knowledge** of the encrypted plaintext.
- In other words, no function of the plaintext can efficiently be guessed notably better from its ciphertext than without it.

# Semantic Security (2/3)

$\mathbf{Exp}^{b}_{\mathcal{CS},A}$ **(Semantic Security Experiment).**

1. **Generate Public Key.** $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$.
2. **Adversarial Choice of Messages.** $(m_0, m_1, s) \leftarrow A(\mathsf{pk})$.
3. **Guess Message.** Return the first output of $A(\mathsf{E}_{\mathsf{pk}}(m_b), s)$.

# Semantic Security (2/3)

$\mathrm{Exp}^b_{\mathcal{CS},A}$ **(Semantic Security Experiment).**

1. **Generate Public Key.** $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^n)$.
2. **Adversarial Choice of Messages.** $(m_0, m_1, s) \leftarrow A(\mathsf{pk})$.
3. **Guess Message.** Return the first output of $A(\mathsf{E}_{\mathsf{pk}}(m_b), s)$.

**Definition.** A cryptosystem $\mathcal{CS} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is said to be **semantically secure** if for every polynomial time algorithm $A$

$$|\Pr[\mathrm{Exp}^0_{\mathcal{CS},A} = 1] - \Pr[\mathrm{Exp}^1_{\mathcal{CS},A} = 1]|$$

is negligible.

# Semantic Security (3/3)

Every semantically secure cryptosystem must be probabilistic!

# Semantic Security (3/3)

Every semantically secure cryptosystem must be probabilistic!

**Theorem.** Suppose that $\mathcal{CS} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is a semantically secure cryptosystem.

Then the related cryptosystem where a $t(n)$-list of messages, with $t(n)$ polynomial, is encrypted by **repeated independent encryption** of each component using the **same public key** is also semantically secure.

# Semantic Security (3/3)

Every semantically secure cryptosystem must be probabilistic!

**Theorem.** Suppose that $\mathcal{CS} = (\mathsf{Gen}, \mathsf{E}, \mathsf{D})$ is a semantically secure cryptosystem.

Then the related cryptosystem where a $t(n)$-list of messages, with $t(n)$ polynomial, is encrypted by **repeated independent encryption** of each component using the **same public key** is also semantically secure.

**Semantic security is useful!**

# The RSA Assumption

**Definition.** The RSA assumption states that if:

1. $N = pq$ factors into two randomly chosen primes $p$ and $q$ of the same bit-size,
2. $e$ is in $\mathbb{Z}^*_{\phi(N)}$,
3. $m$ is randomly chosen in $\mathbb{Z}^*_N$,

then for every polynomial time algorithm $A$

$$\Pr[A(N, e, m^e \bmod N) = m]$$

is negligible.

# Semantically Secure ROM-RSA (1/2)

Suppose that $f : \{0,1\}^n \rightarrow \{0,1\}^n$ is a randomly chosen function (a random oracle).

- **Key Generation.** Choose a random RSA key pair $((N, e), (p, q, d))$, with $\log_2 N = n$.
- **Encryption.** Encrypt a plaintext $m \in \{0,1\}^n$ by choosing $r \in \mathbb{Z}_N^*$ randomly and computing

$$(u, v) = (r^e \bmod N, f(r) \oplus m) \ .$$

- **Decryption.** Decrypt a ciphertext $(u, v)$ by

$$m = v \oplus f(u^d) \ .$$

# Semantically Secure RSA in the ROM (2/2)

- We increase the ciphertext size by a factor of two.
- Our analysis is in the random oracle model, **which is unsound!**

# Semantically Secure RSA in the ROM (2/2)

- We increase the ciphertext size by a factor of two.
- Our analysis is in the random oracle model, **which is unsound!**

**Solutions.**

- Using a "optimal" padding the first problem can be reduced. See standard OAEP+.

# Semantically Secure RSA in the ROM (2/2)

- We increase the ciphertext size by a factor of two.
- Our analysis is in the random oracle model, **which is unsound!**

**Solutions.**

- Using a “optimal” padding the first problem can be reduced. See standard OAEP+.
- Using a scheme with much lower rate, the second problem can be removed.

# Rabin's Cryptosystem (1/3)

**Key Generation.**

- Choose $n$-bit primes $p$ and $q$ such that $p, q = 3 \bmod 4$ randomly and define $N = pq$.
- Output the key pair $(N, (p, q))$, where $N$ is the public key and $(p, q)$ is the secret key.

# Rabin's Cryptosystem (2/3)

**Encryption.** Encrypt a plaintext $m$ by computing

$$c = m^2 \bmod N \enspace .$$

**Decryption.** Decrypt a ciphertext $c$ by computing

$$m = \sqrt{c} \bmod N \enspace .$$

# Rabin's Cryptosystem (2/3)

**Encryption.** Encrypt a plaintext $m$ by computing

$$c = m^2 \bmod N \enspace .$$

**Decryption.** Decrypt a ciphertext $c$ by computing

$$m = \sqrt{c} \bmod N \enspace .$$

There are **four** roots, so which one should be used?

## Rabin's Cryptosystem (3/3)

Suppose $y$ is a quadratic residue modulo $p$.

$$\begin{aligned}\left(\pm y^{(p+1)/4}\right)^2 &= y^{(p+1)/2} \bmod p\\ &= y^{(p-1)/2} y \bmod p\\ &= \left(\frac{y}{p}\right) y\\ &= y \bmod p\end{aligned}$$

In Rabin's cryptosystem:

- Find roots for $y_p = y \bmod p$ and $y_q = y \bmod q$.
- Combine roots to get the four roots modulo $N$. Choose the "right" root and output the plaintext.

# Security of Rabin's Cryptosystem

**Theorem.** Breaking Rabin's cryptosystem is equivalent to factoring.

**Idea.**

1. Choose random element $r$.
2. Hand $r^2 \bmod N$ to adversary.
3. Consider outputs $r'$ from the adversary such that $(r')^2 = r^2 \bmod N$. Then $r' \neq \pm r \bmod N$, with probability $1/2$, in which case $\gcd(r' - r, N)$ gives a factor of $N$.

# A Goldwasser-Micali Variant of Rabin

**Theorem [CG98].** If factoring is hard and $r$ is a random quadratic residue modulo $N$, then for every polynomial time algorithm $A$

$$\Pr[A(N, r^2 \bmod N) = \mathrm{lsb}(r)]$$

is negligible.

- **Encryption.** Encrypt a plaintext $m \in \{0, 1\}$ by choosing a random quadratic residue $r$ modulo $N$ and computing

$$(u, v) = (r^2 \bmod N, \mathrm{lsb}(r) \oplus m) \enspace .$$

- **Decryption.** Decrypt a ciphertext $(u, v)$ by

$$m = v \oplus \mathrm{lsb}(\sqrt{u}) \quad \text{where } \sqrt{u} \text{ is a quadratic residue} \enspace .$$

# Diffie-Hellman Key Exchange (1/3)

Diffie and Hellman asked themselves:

How can two parties efficiently agree on a secret key using only **public** communication?

# Diffie-Hellman Key Exchange (2/3)

**Construction.**
Let $G$ be a cyclic group of order $q$ with generator $g$.

1.
   - Alice picks $a \in \mathbb{Z}_q$ randomly, computes $y_a = g^a$ and hands $y_a$ to Bob.
   - Bob picks $b \in \mathbb{Z}_q$ randomly, computes $y_b = g^b$ and hands $y_b$ to Alice.
2.
   - Alice computes $k = y_b^a$.
   - Bob computes $k = y_a^b$.
3. The joint secret key is $k$.

# Diffie-Hellman Key Exchange (3/3)

**Problems.**

- Susceptible to man-in-the-middle attack without authentication.
- How do we map a random element $k \in G$ to a random symmetric key in $\{0, 1\}^n$?

# The El Gamal Cryptosystem (1/2)

**Definition.** Let $G$ be a cyclic group of order $q$ with generator $g$.

- The **key generation** algorithm chooses a random element $x \in \mathbb{Z}_q$ as the private key and defines the public key as
$$y = g^x \ .$$
- The **encryption** algorithm takes a message $m \in G$ and the public key $y$, chooses $r \in \mathbb{Z}_q$, and outputs the pair
$$(u, v) = \mathrm{E}_y(m, r) = (g^r, y^r m) \ .$$
- The **decryption** algorithm takes a ciphertext $(u, v)$ and the secret key and outputs
$$m = \mathrm{D}_x(u, v) = vu^{-x} \ .$$

## The El Gamal Cryptosystem (2/2)

- El Gamal is essentially Diffie-Hellman + OTP.
- Homomorphic property (with public key $y$)

$$\mathrm{E}_y(m_0, r_0)\mathrm{E}_y(m_1, r_1) = \mathrm{E}_y(m_0 m_1, r_0 + r_1) \ .$$

This property is very important in the construction of cryptographic protocols!

# Discrete Logarithm (1/2)

**Definition.** Let $G$ be a cyclic group of order $q$ and let $g$ be a generator $G$. The **discrete logarithm** of $y \in G$ in the basis $g$ (written $\log_g y$) is defined as the unique $x \in \{0, 1, \ldots, q-1\}$ such that

$$y = g^x \enspace .$$

Compare with a “normal” logarithm! ($\ln y = x$ iff $y = e^x$)

# Discrete Logarithm (2/2)

**Example.** 7 is a generator of $\mathbb{Z}_{12}$ additively, since $\gcd(7, 12) = 1$.

What is $\log_7 3$?

# Discrete Logarithm (2/2)

**Example.** 7 is a generator of $\mathbb{Z}_{12}$ additively, since $\gcd(7, 12) = 1$.

What is $\log_7 3$? ($9 \cdot 7 = 63 = 3 \bmod 12$, so $\log_7 3 = 9$)

# Discrete Logarithm (2/2)

**Example.** 7 is a generator of $\mathbb{Z}_{12}$ additively, since $\gcd(7, 12) = 1$.

What is $\log_7 3$? ($9 \cdot 7 = 63 = 3 \bmod 12$, so $\log_7 3 = 9$)

**Example.** 7 is a generator of $\mathbb{Z}_{13}^*$.

What is $\log_7 9$?

# Discrete Logarithm (2/2)

**Example.** 7 is a generator of $\mathbb{Z}_{12}$ additively, since $\gcd(7, 12) = 1$.

What is $\log_7 3$? $(9 \cdot 7 = 63 = 3 \bmod 12$, so $\log_7 3 = 9)$

**Example.** 7 is a generator of $\mathbb{Z}_{13}^*$.

What is $\log_7 9$? $(7^4 = 9 \bmod 13$, so $\log_7 9 = 4)$

# Discrete Logarithm Assumption

Let $G_{q_n}$ be a cyclic group of prime order $q_n$ such that $\lfloor \log_2 q_n \rfloor = n$ for $n = 2, 3, 4, \ldots$, and denote the family $\{G_{q_n}\}_{n\in\mathbb{N}}$ by $G$.

**Definition.** The **Discrete Logarithm (DL) Assumption** in $G$ states that if generators $g_n$ and $y_n$ of $G_{q_n}$ are randomly chosen, then for every polynomial time algorithm $A$

$$\Pr\left[A(g_n, y_n) = \log_{g_n} y_n\right]$$

is negligible.

# Discrete Logarithm Assumption

Let $G_{q_n}$ be a cyclic group of prime order $q_n$ such that $\lfloor \log_2 q_n \rfloor = n$ for $n = 2, 3, 4, \ldots$, and denote the family $\{G_{q_n}\}_{n \in \mathbb{N}}$ by $G$.

**Definition.** The **Discrete Logarithm (DL) Assumption** in $G$ states that if generators $g$ and $y$ of $G$ are randomly chosen, then for every polynomial time algorithm $A$

$$\Pr\left[A(g, y) = \log_g y\right]$$

is negligible.

We usually remove the indices from our notation!