Cryptography

Computational Security

Def.:

Encriptionscheme Σ uniform ciphertexts when:

$$L_{OTS\$-real} \equiv L_{OTS\$-rand}$$

$$\begin{array}{c} L_{OTS\$-real} \\ cTXT(m): \\ k \leftarrow \Sigma.KeyGen() \\ c \leftarrow \Sigma.Enc(k,m) \\ return\ c \end{array} \qquad \begin{array}{c} L_{OTS\$-rand} \\ cTXT(m): \\ c \leftarrow \Sigma.C \\ return\ c \end{array}$$

Def.:

Encriptionscheme Σ has one-time secrecy when:

$$L_{OTS-L} \equiv L_{OTS-R}$$

$$\begin{array}{c} L_{OTS-L} \\ Eavesdrop(m_L, m_R): \\ k \leftarrow \Sigma.KeyGen() \\ c \leftarrow \Sigma.Enc(k,m_L) \\ return\ c \end{array} \qquad \begin{array}{c} L_{OTS-R} \\ Eavesdrop(m_L, m_R): \\ k \leftarrow \Sigma.KeyGen() \\ c \leftarrow \Sigma.Enc(k,m_R) \\ return\ c \end{array}$$

Composing libraries

Theorem:

If $L_L \equiv L_R$ then for all $L^\star$:

$$L^\star \diamond L_L \equiv L^\star \diamond L_R$$

Pf.:

A arbitrary program that accesses $L^\star$

$$P[A \diamond (L^\star \diamond L_L) \Rightarrow 1] = P[\underbrace{(A \diamond L^\star)}_{A^\star} \diamond L_L \Rightarrow 1] = P[(A \diamond L^\star) \diamond L_R \Rightarrow 1] = P[A \diamond (L^\star \diamond L_R) \Rightarrow 1]$$

Theorem:

Σ with uniform ciphertexts (OTS$) also has one-time secrecy (OTS)

$$\begin{aligned} L_{OTS\$-real} &\equiv L_{OTS\$-rand} \\ \Rightarrow L_{OTS-L} &\equiv L_{OTS-R} \end{aligned}$$

- construct sequence of libraries, s.t. each two are interchangeable
- We call the intermediate ones "hybrids"
- Exploit $\equiv$ interchangeable

Proof

$$L_{hyb1}$$

$$\begin{array}{l} L_{OTS-L} \\ \; Eavesdrop(m_L, m_R): \\ \;\; k \leftarrow \Sigma.KeyGen() \\ \;\; c \leftarrow \Sigma.Enc(k, m_L) \\ \;\; return\ c \end{array} \equiv \begin{array}{l} L \\ \; Eavesdrop(m_L, m_R): \\ \;\; c := cTXT(m_L) \\ \;\; return\ c \end{array} \Diamond \begin{array}{l} L_{OTS\$-real} \\ \; cTXT(m): \\ \;\; k \leftarrow \Sigma.KeyGen() \\ \;\; c \leftarrow \Sigma.Enc(k, m) \\ \;\; return\ c \end{array}$$

$$L_{hyb2}$$

$$\begin{array}{l} L_{OTS-L} \\ \; Eavesdrop(m_L, m_R): \\ \;\; k \leftarrow \Sigma.KeyGen() \\ \;\; c \leftarrow \Sigma.Enc(k, m_L) \\ \;\; return\ c \end{array} \equiv \begin{array}{l} L \\ \; Eavesdrop(m_L, m_R): \\ \;\; c := cTXT(m_L) \\ \;\; return\ c \end{array} \Diamond \begin{array}{l} L_{OTS\$-rand} \\ \; cTXT(m): \\ \;\; c \leftarrow \Sigma.C \\ \;\; return\ c \end{array}$$

$$L_{hyb3}$$

$$\begin{array}{l} L_{OTS-L} \\ \; Eavesdrop(m_L, m_R): \\ \;\; k \leftarrow \Sigma.KeyGen() \\ \;\; c \leftarrow \Sigma.Enc(k, m_L) \\ \;\; return\ c \end{array} \equiv \begin{array}{l} L \\ \; Eavesdrop(m_L, m_R): \\ \;\; c := cTXT(m_R) \\ \;\; return\ c \end{array} \Diamond \begin{array}{l} L_{OTS\$-rand} \\ \; cTXT(m): \\ \;\; c \leftarrow \Sigma.C \\ \;\; return\ c \end{array}$$

$$L_{hyb4}$$

$$\begin{array}{l} L_{OTS-L} \\ \; Eavesdrop(m_L, m_R): \\ \;\; k \leftarrow \Sigma.KeyGen() \\ \;\; c \leftarrow \Sigma.Enc(k, m_L) \\ \;\; return\ c \end{array} \equiv \begin{array}{l} L \\ \; Eavesdrop(m_L, m_R): \\ \;\; c := cTXT(m_R) \\ \;\; return\ c \end{array} \Diamond \begin{array}{l} L_{OTS\$-real} \\ \; cTXT(m): \\ \;\; k \leftarrow \Sigma.KeyGen() \\ \;\; c \leftarrow \Sigma.Enc(k, m) \\ \;\; return\ c \end{array}$$

$$\begin{array}{l} L_{OTS-L} \\ \; Eavesdrop(m_L, m_R): \\ \;\; k \leftarrow \Sigma.KeyGen() \\ \;\; c \leftarrow \Sigma.Enc(k, m_L) \\ \;\; return\ c \end{array} \equiv \begin{array}{l} L_{OTS-R} \\ \; Eavesdrop(m_L, m_R): \\ \;\; k \leftarrow \Sigma.KeyGen() \\ \;\; c \leftarrow \Sigma.Enc(k, m_R) \\ \;\; return\ c \end{array}$$

4. Security from intractable computations

Intractable computations

What is tractable?
Everything computable probabilistic polynomial time.
Computational security is based on difficulty of intractable computation.
Modern Cryptology is based on computational security models.

Security parameter λ

λ quantities the computational effort of an attack and the security of an algorithm.
Iddeally we would like the cost of an attack to be $\sim\ 2^{\lambda}$ and cost of implementation $\sim\ poly(\lambda)$ [even better: $\sim \lambda$].

Examples:

AES-128 uses 128-bit keys an attacker would need to perform 2^{128} operations.

Formalizing "efficiently"

- According to complexity theory, we consider everything computable in probabilistic polynomial time (ppt) to be efficient.

- TM halts after p(λ) steps, where p($\bullet$) polynomial.

Examples:

λ^2	*polynomial*
$2^{(log\ \lambda)^{10}}$	*superpolynomial*
1.1^{λ}	*exponential*
λ^{1000}	*polynomial*

Formalizing "negligible probabilities"

- Adversary can always guess our λ-bit secret key with probability $2^{-\lambda}$
- Adversary can repeat a guess $p(\lambda)$ times (for some polynomial p(•)) and achieve *probability* $\leq p(\lambda) \cdot 2^{-\lambda}$ .... should still be negligible

Definition:

A function f() is negligible iff for all polynomials p(•):

$$\lim_{\lambda \to \infty} p(\lambda) \cdot f(\lambda) = 0$$

Definition:

A function g() is superpolynomial when $\frac{1}{g()}$ is negligible.

Definition:

For functions p() dand q() we write:

$$p \approx q$$
$$for \; \underbrace{\mid p - q \mid}_{|p(\lambda) - q(\lambda)|} \; is\ negligible$$

Extending the notion to probabilities

$$\begin{aligned} P[\epsilon] \approx 0 &: Event\ \epsilon\ almost\ never\ occurs \\ P[\epsilon] \approx 1 &: Event\ \epsilon\ occurs\ almost\ surely/almost\ certainly/with\ overwhelming\ probability \\ P[A] \approx P[B] &: A\ and\ B\ have\ essentially\ the\ same\ probability \end{aligned}$$

Indistinguishability

- Interchangeable libraries produce the same probability distributions
- Indistinguishable libraries produce essentially the same probability distributions