# Introduction to Modern Cryptography: Revision[1]

Giorgos Panagiotakos

Course Organizer: Prof. Aggelos Kiayias

May 18, 2018

[1] The slides are based on the course notes.

# Commitments

## Syntax

- $b \leftarrow Param(1^{\lambda})$;
- $(r, c) \leftarrow Commit(b, M)$;
- $True/False \leftarrow Verify(b, c, r, M)$

# Commitments

## Syntax

- $b \leftarrow Param(1^{\lambda})$;
- $(r, c) \leftarrow Commit(b, M)$;
- $True/False \leftarrow Verify(b, c, r, M)$

## Security Properties

- Correctness;
- Binding;
- Hiding

# Commitments

## Syntax

- $b \leftarrow Param(1^\lambda)$;
- $(r, c) \leftarrow Commit(b, M)$;
- $True/False \leftarrow Verify(b, c, r, M)$

## Security Properties

- Correctness;
- Binding;
- Hiding

*Correctness*: For every message $M$:

$$\Pr\left[\begin{array}{l} b \leftarrow Param(1^\lambda); (r, c) \leftarrow Commit(b, M) : \\ Verify(b, c, r, M) = True \end{array}\right] = 1$$

# Binding

The committer should not be able to de-commit to a value different from the one she committed.

**Algorithm 1** $bindattack_{\mathcal{A}}(1^{\lambda})$

1: Let $b \leftarrow Param(1^{\lambda})$
2: $(c, r_1, M_1, r_2, M_2) \leftarrow \mathcal{A}(1^{\lambda}, b)$
3: **if** $M_1 \neq M_2$ **and** $Ver(b, c, r_1, M_1) = 1$ **and** $Ver(b, c, r_2, M_2) = 1$ **then**
4: **return** 1
5: **else**
6: **return** 0
7: **end if**

# Binding

The committer should not be able to de-commit to a value different from the one she committed.

**Algorithm 2** $bindattack_{\mathcal{A}}(1^{\lambda})$

1: Let $b \leftarrow Param(1^{\lambda})$
2: $(c, r_1, M_1, r_2, M_2) \leftarrow \mathcal{A}(1^{\lambda}, b)$
3: **if** $M_1 \neq M_2$ **and** $Ver(b, c, r_1, M_1) = 1$ **and** $Ver(b, c, r_2, M_2) = 1$ **then**
4: **return** 1
5: **else**
6: **return** 0
7: **end if**

$$\forall \text{ PPT } \mathcal{A} : \Pr[bindattack_{\mathcal{A}}(1^{\lambda}) = 1] = negl(\lambda)$$

# Hiding

No information is leaked about the message before de-commitment.

**Algorithm 3** $hidingattack_{\mathcal{A}}(1^{\lambda})$

$b \leftarrow Param(1^{\lambda})$
$(aux, M_0, M_1) \leftarrow \mathcal{A}_1(1^{\lambda}, b)$
$d \xleftarrow{R} \{0, 1\}$
$(r, c) \leftarrow Commit(b, M_d)$
$d^* \leftarrow \mathcal{A}_2(c, b, aux)$
**if** $d^* = d$ and $M_0 \neq M_1$ **then**
    **return** 1
**else**
    **return** 0
**end if**

# Hiding

No information is leaked about the message before de-commitment.

**Algorithm 4** $hidingattack_{\mathcal{A}}(1^{\lambda})$

1: $b \leftarrow Param(1^{\lambda})$
2: $(aux, M_0, M_1) \leftarrow \mathcal{A}_1(1^{\lambda}, b)$
3: $d \xleftarrow{R} \{0,1\}$
4: $(r, c) \leftarrow Commit(b, M_d)$
5: $d^* \leftarrow \mathcal{A}_2(c, b, aux)$
6: **if** $d^* = d$ and $M_0 \neq M_1$ **then**
7: **return** 1
8: **else**
9: **return** 0
10: **end if**

$$\forall \text{ PPT } \mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2) : \Pr[hidingattack_{\mathcal{A}}(1^{\lambda}) = 1] \leq \frac{1}{2} + negl(\lambda)$$

# Key Exchange

## Syntax

For parties $A, B$:

- $trans_{A,B}(1^{\lambda})$;
- $key(\tau)$

# Key Exchange

## Syntax

For parties $A, B$:

- $trans_{A,B}(1^\lambda)$;
- $key(\tau)$

## Security Definition

Let $V$ be some PT predicate such that

$$\delta = \Pr_{\tau \leftarrow trans_{A,B}(1^\lambda)}[V(key(\tau)) = 1]$$

# Key Exchange

## Syntax

For parties $A, B$:

- $trans_{A,B}(1^\lambda)$;
- $key(\tau)$

## Security Definition

Let $V$ be some PT predicate such that

$$\delta = \Pr_{\tau \leftarrow trans_{A,B}(1^\lambda)}[V(key(\tau)) = 1]$$

For any PPT $\mathcal{A}$ it holds that:

$$\Pr_{\tau \leftarrow trans_{A,B}(1^\lambda)}[\mathcal{A}(\tau) = V(key(\tau)] \leq \max\{\delta, 1-\delta\} + negl(\lambda)$$

# Zero Knowledge Proofs

## Syntax

For interactive programs $\langle P, V \rangle$:

- language $L \in NP$, $R$ polynomial time predicate such that

$$L = \{x : R(x, w) = 1 \text{ for some } w\}$$

- $out^P_{P,V}(x, w, z)$

# Zero Knowledge Proofs

## Syntax

For interactive programs $\langle P, V \rangle$:

- language $L \in NP$, $R$ polynomial time predicate such that

$$L = \{x : R(x, w) = 1 \text{ for some } w\}$$

- $out^P_{P,V}(x, w, z)$

## Security Properties

- Completeness;
- Soundness;
- Zero-Knowledge

# Zero Knowledge Proofs

## Syntax

For interactive programs $\langle P, V \rangle$:

- language $L \in NP$, $R$ polynomial time predicate such that

$$L = \{x : R(x, w) = 1 \text{ for some } w\}$$

- $out^P_{P,V}(x, w, z)$

## Security Properties

- Completeness;
- Soundness;
- Zero-Knowledge

*Completeness*: If $x \in L$ and $R(x, w) = 1$ for some $w$, then for all strings $z$: $\Pr[out^V_{P,V}(x, w, z) = 1] \geq 1 - negl(\lambda)$

# Soundness

If the verifier is convinced, then we can efficiently extract a witness for $x$.

## Definition

For any PPT $\mathcal{P}^*$ and arbitrary $x, w, z$ let

$$\pi_{x,w,z} = \Pr[out^{\mathcal{V}}_{\mathcal{P}^*,\mathcal{V}}(x, w, z) = 1].$$

$\langle P, V \rangle$ is sound iff there exists non-negligible $s(\lambda), q(\lambda)$ such that $\forall$ PPT $\mathcal{P}^*$, $\exists$ PPT $K$ such that if

$$\tilde{\pi}_{x,w,z} = \Pr[K(x, w, z) = w' : R(x, w') = 1]$$

then $\pi_{x,w,z} \geq s(\lambda)$ implies $\tilde{\pi}_{x,w,z} \geq q(\lambda)$.

# (Statistical) Zero-knowledge

The verifier can simulate the whole transcript on his own.

Definition

$\forall$ PPT $\mathcal{V}^*$, $\exists$ PPT S, such that for all $x, w$ with $R(x, w) = 1$:

$$\forall \mathcal{A} |\Pr[\mathcal{A}(S(x, z)) = 1] - \Pr[\mathcal{A}(out^{\mathcal{V}^*}_{\mathcal{P},\mathcal{V}^*}(x, w, z)]| \leq \epsilon$$

# Digital Signatures

## Syntax

- $(vk, sk) \leftarrow Gen(1^{\lambda})$;
- $\sigma \leftarrow Sign(sk, M)$;
- $True/False \leftarrow Verify(vk, M, \sigma)$

# Digital Signatures

## Syntax

- $(vk, sk) \leftarrow Gen(1^{\lambda})$;
- $\sigma \leftarrow Sign(sk, M)$;
- $True/False \leftarrow Verify(vk, M, \sigma)$

## Security Properties

- Correctness;
- Unforgeability

# Digital Signatures

## Syntax

- $(vk, sk) \leftarrow Gen(1^\lambda)$;
- $\sigma \leftarrow Sign(sk, M)$;
- $True/False \leftarrow Verify(vk, M, \sigma)$

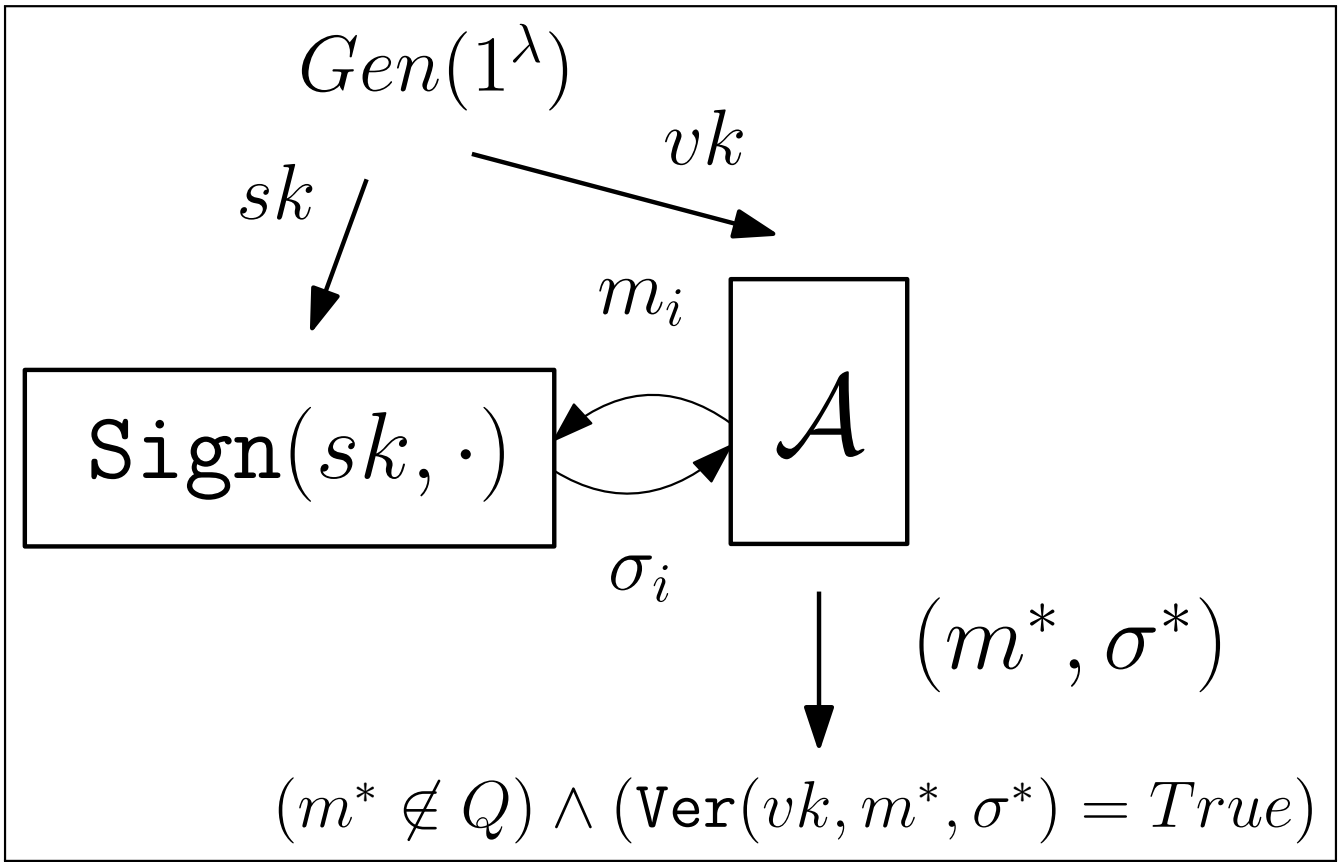
## Security Properties

- Correctness;
- Unforgeability

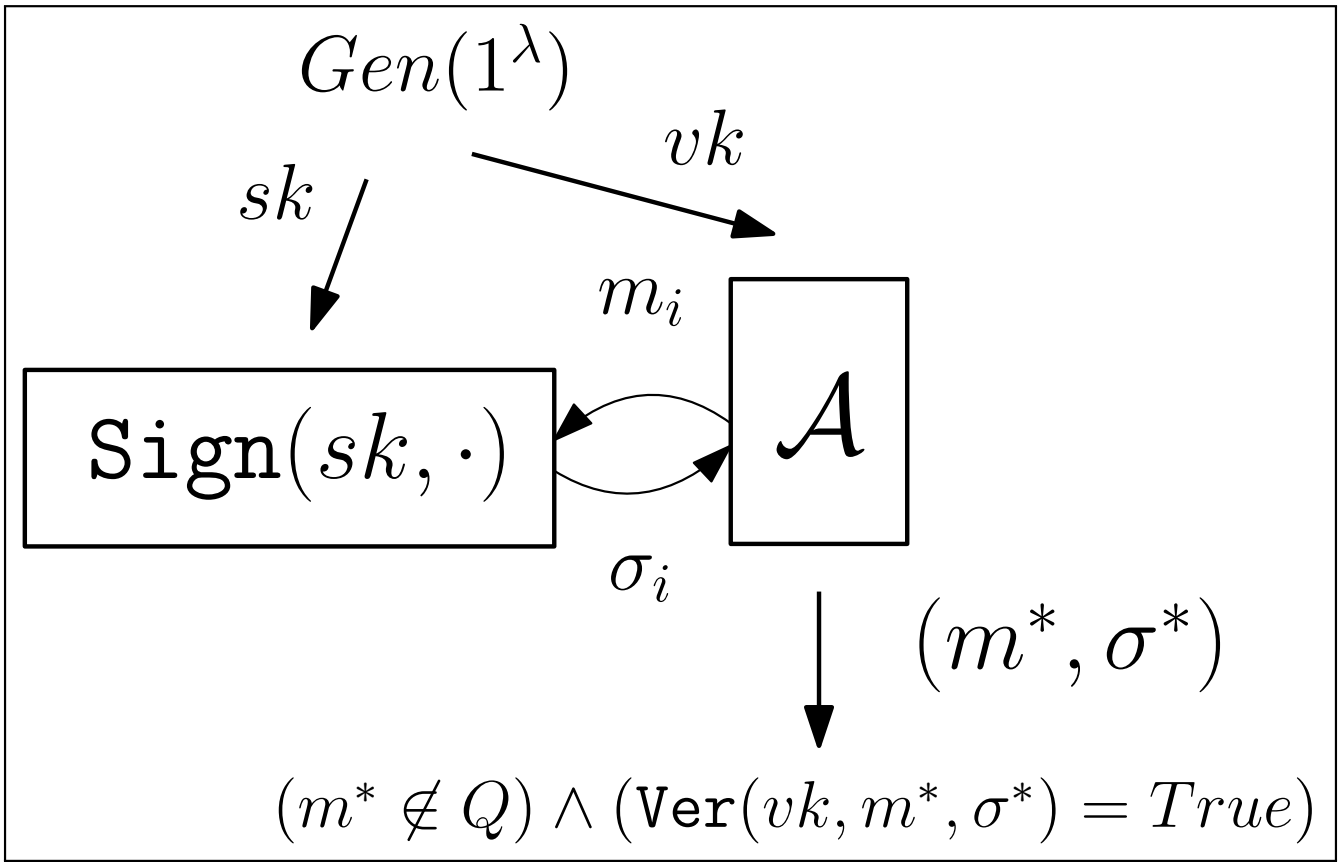
*Correctness*: For every $(vk, sk) \in Gen(1^\lambda)$, every $M \in \{0,1\}^*$:

$$\Pr\left[ Verify(vk, M, Sign(sk, M)) = True \right] = 1$$

# Unforgeability (UF-CMA)



# Unforgeability (UF-CMA)




$$\forall \text{ PPT } \mathcal{A} : \Pr[Exp_{\mathcal{A}}^{UF}(1^\lambda) = 1] \leq negl(\lambda)$$

# Public-key Encryption

## Syntax

- $(pk, sk) \leftarrow Gen(1^{\lambda})$;
- $c \leftarrow Enc(pk, M)$;
- $m \leftarrow Dec(sk, c)$

# Public-key Encryption

## Syntax

- $(pk, sk) \leftarrow Gen(1^{\lambda})$;
- $c \leftarrow Enc(pk, M)$;
- $m \leftarrow Dec(sk, c)$

## Security Properties

- Correctness;
- IND-CPA

# Public-key Encryption

## Syntax

- $(pk, sk) \leftarrow Gen(1^\lambda)$;
- $c \leftarrow Enc(pk, M)$;
- $m \leftarrow Dec(sk, c)$

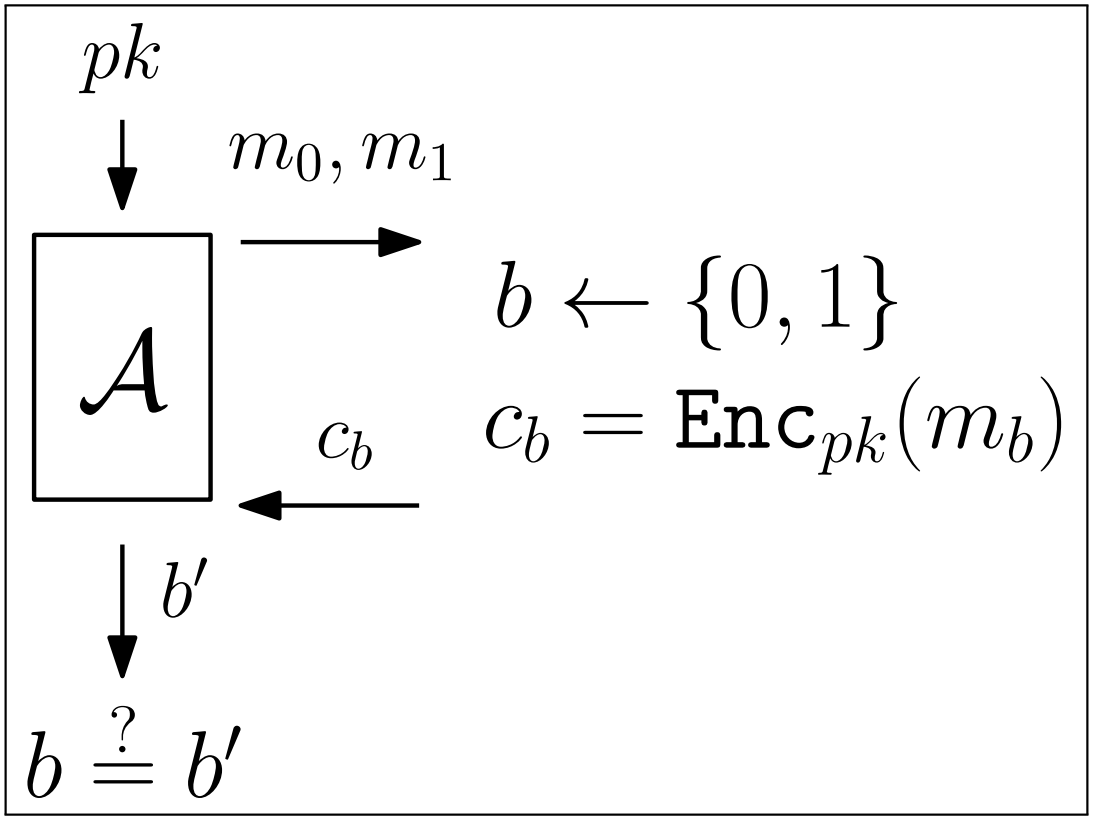
## Security Properties

- Correctness;
- IND-CPA

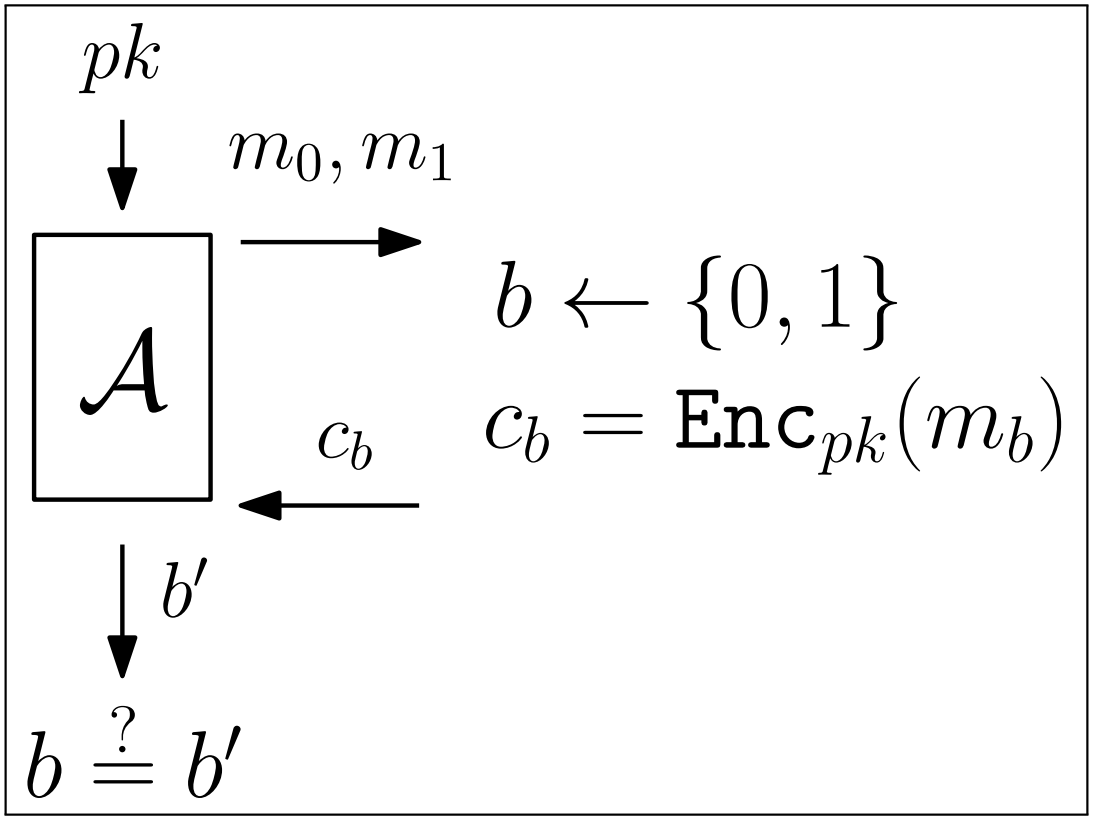
*Correctness*: For every $(pk, sk) \in Gen(1^\lambda)$, every $M \in \{0,1\}^*$:

$$\Pr\left[\, Dec(sk, Enc(pk, M)) = M \quad \right] = 1$$

# IND-CPA


$pk$
$m_0, m_1$
$\mathcal{A}$
$b \leftarrow \{0,1\}$
$c_b = \mathtt{Enc}_{pk}(m_b)$
$c_b$
$b'$
$b \stackrel{?}{=} b'$

# IND-CPA




$\forall$ PPT $\mathcal{A}$ : $\Pr[Exp_{\mathcal{A}}^{IND-CPA}(1^{\lambda}) = 1] \leq 1/2 + negl(\lambda)$

# Questions?