SIGNATURES IN THE PRESENCE OF KEY-DEPENDENT MESSAGES

Madeline González Muñiz
(joint work with Rainer Steinwandt)

KDM-CPA Security

- Motivation: encrypting a hard drive which has the secret encryption key stored
- Adversary queries an encryption of a function of the secret key
- The goal is indistinguishability between encryptions of the secret key and random messages of the same length

Signature Schemes

- Signing a backup of a hard disk which has the signing key stored
- Combined public key schemes
- Digital Signature Scheme Algorithms:
  - Key Generation
  - Signing Algorithm
  - Verification Algorithm

Key Generation

- K is a probabilistic key generation algorithm
- K returns a pair of keys (sk, pk)
- We interpret sk as the state of the signer

Signing Algorithm

- S is a signing algorithm
- On input M and state returns a signature σ or an error symbol ⊥.
- Moreover, the state value sk is updated.

Verification Algorithm

- V is a deterministic verification algorithm
- On input a public key pk, a message M, and a candidate signature σ for M returns 1 or 0.

EUF-CMA Security

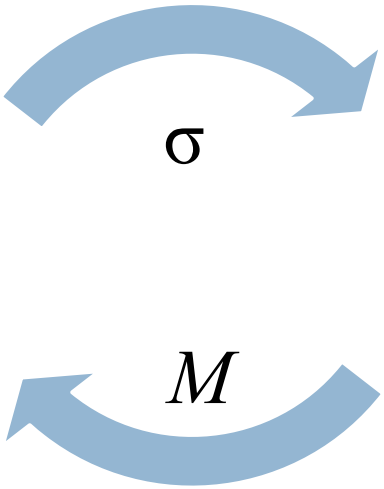
- Let (K, S, V) be a signature scheme and A an adversary

Run K and generate (sk, pk)

pk

EUF-CMA Security

In the one-time EUF-CMA setting, A gets only one query



EUF-CMA Security

M, σ

EUF-CMA Security

- A wins if…
  - $V_{pk}(M, \sigma)=1$
  - M was not queried earlier

(M, σ)

KDS-CMA Security

Now the adversary can query a function g of the secret key

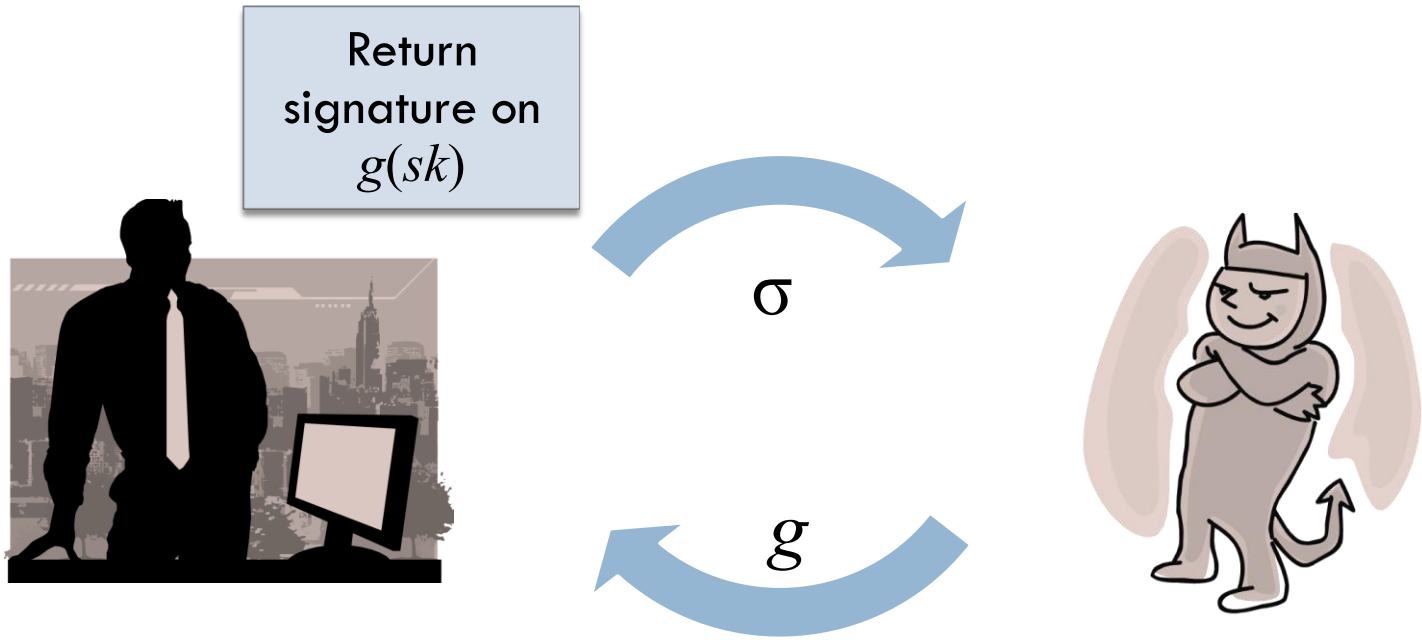
KDS-CMA Security

- Let (K, S, V) be a signature scheme and A an adversary

Run K and generate (sk, pk)

pk

KDS-CMA Security



KDS-CMA Security

M, σ

KDS-CMA Security

- A wins if…
  - $V_{pk}(M, \sigma)=1$
  - M was not queried earlier (that is, $M \neq g(sk)$ for any query)

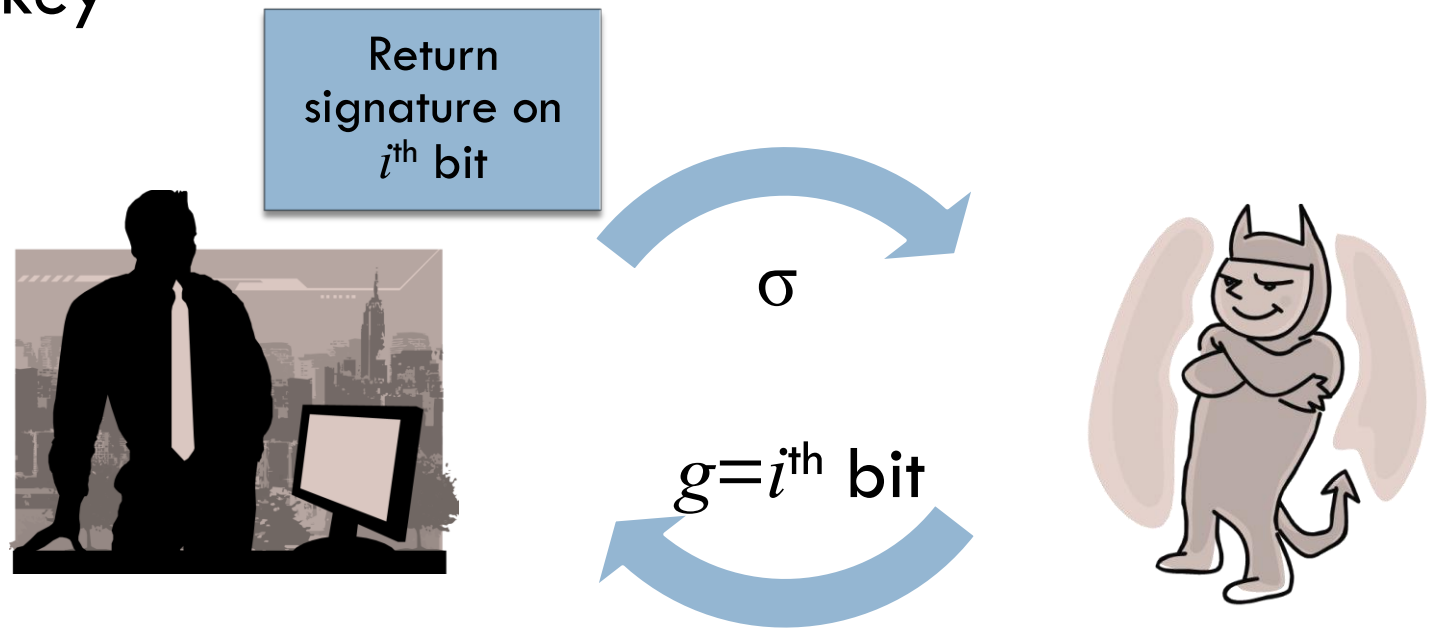
(M, σ)

Impossibility of Stateless Scheme

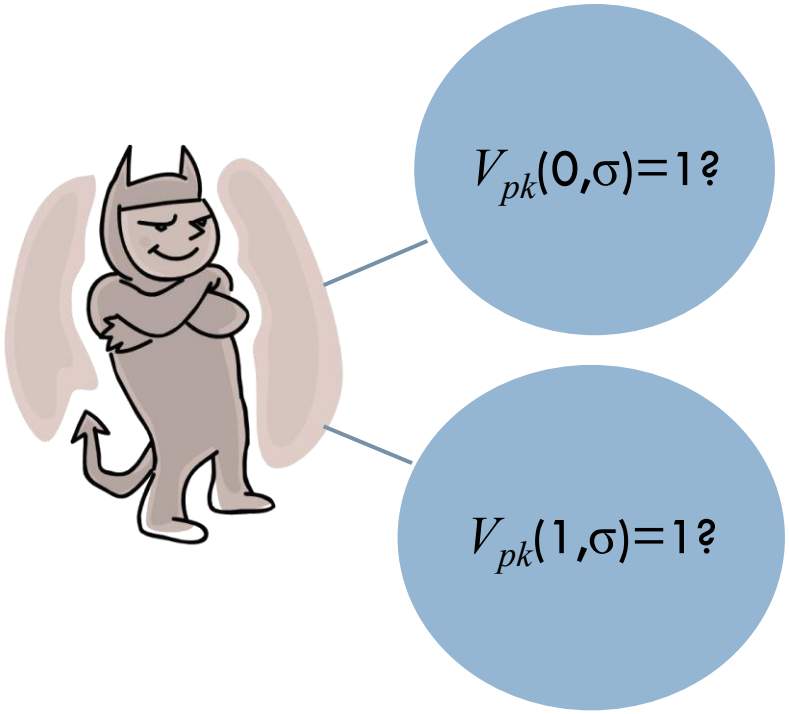
The secret key must be necessarily updated in this scenario

Impossibility of Stateless Scheme

- Let $sk=b_0 \ldots b_{k-1}$ be the bit representation of a k-bit key



Impossibility of Stateless Scheme



After k queries, A has the entire key, and this attack generalizes to a polynomial number of fixed keys.

Forward Security

- Compromise of the secret key does not enable an adversary to forge signatures pertaining to the past
- In addition to (K, S, V), there is an algorithm U which updates the secret key
- We adapt a forward secure scheme to build one secure in the KDS-CMA sense

Long Signatures

- Key Generation
  - Create pair (sk_0, pk_0)
  - $sk=(sk_0, [])$ for empty list []
    $pk=pk_0$
- Key Update
  - Create fresh key pair (sk_i, pk_i)
  - $\sigma_i=S_{sk_{i-1}}(i \mid pk_i)$
  - $Cert_i=(pk_i, \sigma_i)$
  - New state is $(sk_i, Certs)$

Long Signatures

- Signing Algorithm
  - Let $s = S_{sk_i}(M)$
  - Return $(s, Certs)$
- Verification Algorithm
  - $V_{pk_i}(M,s)=1$
  - $V_{pk_{j-1}}(pk_j, \sigma_j)=1$ for $j=1 \ldots i$

Long Signatures are Insecure

This σ is used to create $Cert_{i+1}$ which A can use to forge by setting $sk_{i+1}=sk^*$

Create (sk^*, pk^*) during period i

σ

$M=(i+1)\,|\,pk^*$

One-Time EUF-CMA to KDS-CMA Compiler

- Let (K, S, V) be a one-time EUF-CMA secure signature scheme
- Key Generation
  - Create pair (sk_0^c, pk_0^c)
  - Will be used to create a certificate chain
  - $sk=(sk_0^c, \lambda, [])$ for empty string λ and empty list $[]$
    $pk=pk_0^c$

One-Time EUF-CMA to KDS-CMA Compiler

Signing Algorithm (i^{th} message)

- Create two fresh key pairs (sk_i^c, pk_i^c) and (sk_i^m, pk_i^m)
- $\sigma_i = S_{sk_{i-1}{}^c}(pk_i^c \mid pk_i^m)$
- $Cert_i = pk_i^c \mid pk_i^m \mid \sigma_i$
- $s = S_{sk_i{}^m}(r \mid H(M \mid r))$
- Signature is $(r, s, Certs)$ and the state is updated

Verification Algorithm

- $V_{pk_i{}^m}(r \mid H(M \mid r), s) = 1$
- $V_{pk_{j-1}{}^c}(pk_j^c \mid pk_j^m, \sigma_j) = 1$ for $j = 1 \ldots i$

Sketch of the Proof

Let B be an EUF-CMA adversary

Let A be a KDS-CMA adversary

If A can forge, B can simulate and also forge

Simulation by B

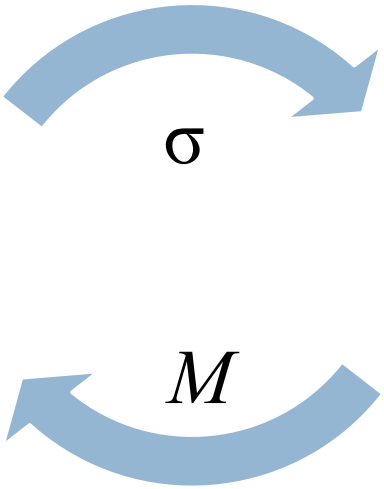
- Let (K, S, V) be a one-time signature scheme secure in the sense of EUF-CMA

Run K and generate (sk^*, pk^*)

pk^*

Simulation by B

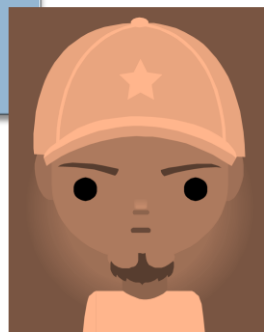
- B gets one signature query under pk^*



Simulation by B

- Let (K', S', V') be the compiled signature scheme

Run K' and generate (sk_0^c, pk_0^c)

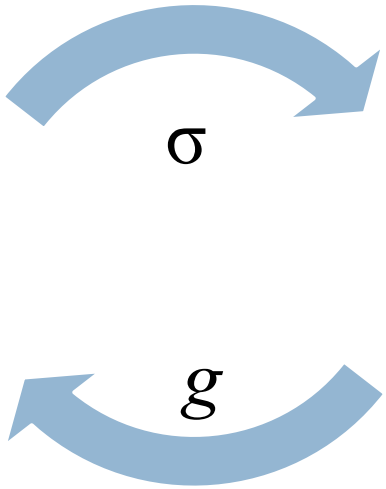


pk_0^c

Simulation by B

Create (sk_i^c, pk_i^c) and (sk_i^m, pk_i^m)

Return signature on $g(sk)$



What about pk^*?

Simulation by B

- Two options:
  - Create (sk_i^c, pk_i^c) and set $pk^* = pk_i^m$
  - Create (sk_i^m, pk_i^m) and set $pk^* = pk_i^c$

Simulation by B

σ

M

If A can guess the output of g with non-negligible probability, we modify A to submit a non key-dependent query.

Return signature on M

σ

$g=M$

Simulation by B

σ

M

What if A cannot guess the output of g with non-negligible probability?

Create (sk', pk') and set $M=g(sk')$

σ

g

Questions?