Constructing Provably-Secure Identity-Based Signature Schemes

Chethan Kamath

Indian Institute of Science, Bangalore

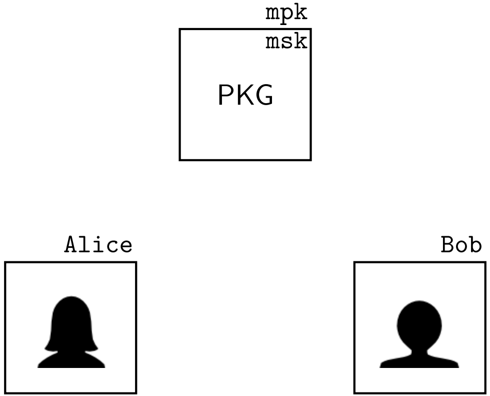
November 23, 2013

Table of contents

Contents

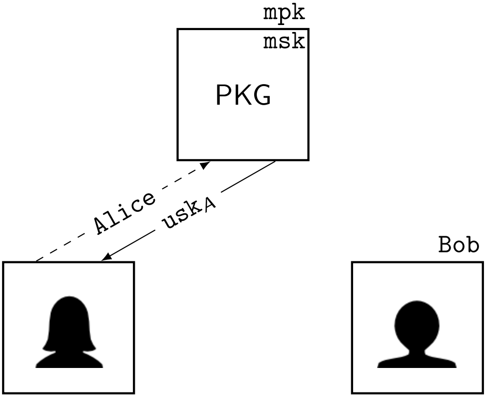
Identity-Based Cryptography

- Introduced by Shamir in 1984.
- Any *arbitrary* string can be used as public key.
- Certificate management can be avoided.
- A trusted *private key generator* (PKG) generates secret keys.



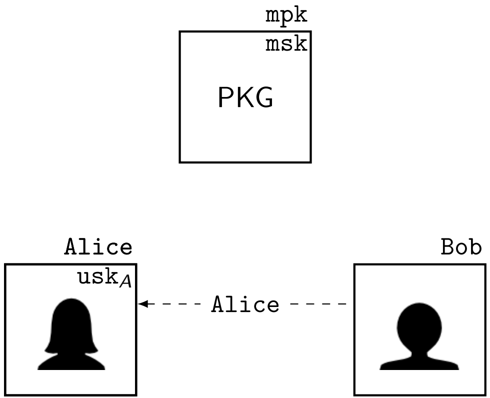
Identity-Based Cryptography

- Introduced by Shamir in 1984.
- Any *arbitrary* string can be used as public key.
- Certificate management can be avoided.
- A trusted *private key generator* (PKG) generates secret keys.



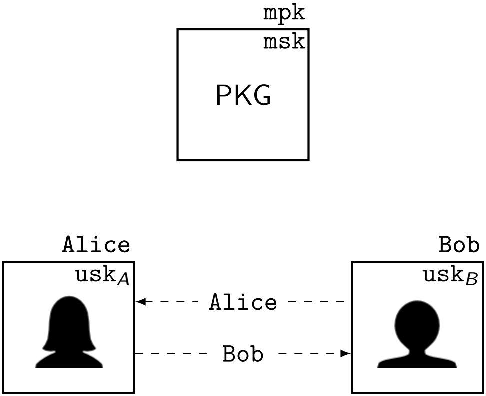
Identity-Based Cryptography

- Introduced by Shamir in 1984.
- Any *arbitrary* string can be used as public key.
- Certificate management can be avoided.
- A trusted *private key generator* (PKG) generates secret keys.



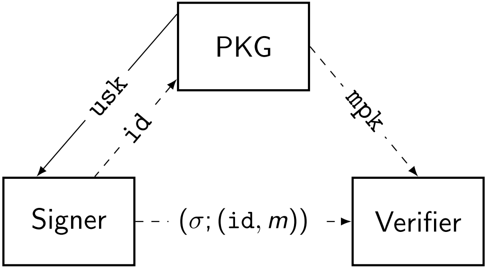
Identity-Based Cryptography

- Introduced by Shamir in 1984.
- Any *arbitrary* string can be used as public key.
- Certificate management can be avoided.
- A trusted *private key generator* (PKG) generates secret keys.



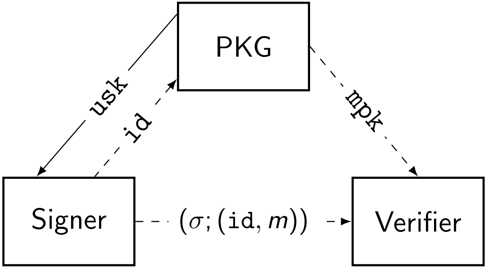
Identity-Based Signatures

- IBS: digital signatures extended to identity-based setting



Identity-Based Signatures

- IBS: digital signatures extended to identity-based setting



- Focus of the work: construction of IBS schemes
  1. **Concrete IBS based on Schnorr signature**
  2. Generic construction from a *weaker* model

Contents

Public-Key Signature

Consists of three PPT algorithms $\{\mathcal{K}, \mathcal{S}, \mathcal{V}\}$:

- **Key Generation**, $\mathcal{K}(\kappa)$
  - Used by the *signer* to generate the key-pair ($\mathtt{pk}$,$\mathtt{sk}$)
  - $\mathtt{pk}$ is published and the $\mathtt{sk}$ kept secret
- **Signing**, $\mathcal{S}_{\mathtt{sk}}(m)$
  - Used by the *signer* to generate signature on some message m
  - The secret key $\mathtt{sk}$ used for signing
- **Verification**, $\mathcal{V}_{\mathtt{pk}}(\sigma, m)$
  - Used by the *verifier* to validate a signature
  - Outputs 1 if σ is a valid signature on m; else, outputs 0

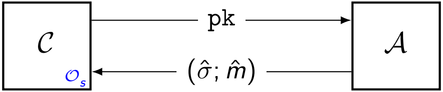
Identity-Based Signature

Consists of four PPT algorithms $\{\mathcal{G}, \mathcal{E}, \mathcal{S}, \mathcal{V}\}$:

- **Set-up**, $\mathcal{G}(\kappa)$
  - Used by *PKG* to generate the master key-pair ($\mathtt{mpk}$,$\mathtt{msk}$)
  - $\mathtt{mpk}$ is published and the $\mathtt{msk}$ kept secret
- **Key Extraction**, $\mathcal{E}_{\mathtt{msk}}(\mathtt{id})$
  - Used by *PKG* to generate the user secret key ($\mathtt{usk}$)
  - $\mathtt{usk}$ is then distributed through a secure channel
- **Signing**, $\mathcal{S}_{\mathtt{usk}}(\mathtt{id}, m)$
  - Used by the *signer* (with identity $\mathtt{id}$) to generate signature on some message m
  - The *user* secret key $\mathtt{usk}$ used for signing
- **Verification**, $\mathcal{V}_{\mathtt{mpk}}(\sigma, \mathtt{id}, m)$
  - Used by the *verifier* to validate a signature
  - Outputs 1 if σ is a valid signature on m by the user with identity $\mathtt{id}$; otherwise, outputs 0

STANDARD SECURITY MODELS

Security Model for PKS: EU-CMA

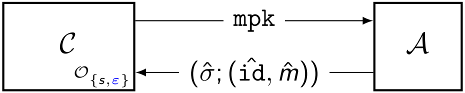


- Existential unforgeability under chosen-message attack
  1. $\mathcal{C}$ generates key-pair $(\texttt{pk}, \texttt{sk})$ and passes $\texttt{pk}$ to $\mathcal{A}$
  2. $\mathcal{A}$ allowed: Signature Queries through an oracle $\mathcal{O}_s$
  3. Forgery: $\mathcal{A}$ wins if $(\hat{\sigma}; \hat{m})$ is *valid* and *non-trivial*

- Adversary's advantage in the game:

$$\Pr\left[1 \leftarrow \mathcal{V}_{\texttt{pk}}(\hat{\sigma}; \hat{m}) : (\texttt{sk}, \texttt{pk}) \xleftarrow{\$} \mathcal{K}(\kappa); (\hat{\sigma}; \hat{m}) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_s}(\texttt{pk})\right]$$

Security Model for IBS: EU-ID-CMA



- Existential unforgeability with adaptive identity under chosen-message attack
  1. $\mathcal{C}$ generates key-pair $(\mathtt{mpk}, \mathtt{msk})$ and passes mpk to $\mathcal{A}$
  2. $\mathcal{A}$ allowed: Signature Queries, Extract Queries
  3. Forgery: $\mathcal{A}$ wins if $(\hat{\sigma}; (\hat{\mathtt{id}}, \hat{m}))$ is *valid* and *non-trivial*

- Adversary's advantage in the game:

$$\Pr\left[1 \leftarrow \mathcal{V}_{\mathtt{mpk}}(\hat{\sigma}; (\hat{\mathtt{id}}, \hat{m})) : (\mathtt{msk}, \mathtt{mpk}) \xleftarrow{\$} \mathcal{G}(\kappa); (\hat{\sigma}; (\hat{\mathtt{id}}, \hat{m})) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\{s,\varepsilon\}}}(\mathtt{mpk})\right]$$

SCHNORR SIGNATURE AND ORACLE REPLAY ATTACK

Schnorr Signature: Features

- Derived from Schnorr identification (FS Transform)
- Uses one hash function
- Security:
  - Based on *discrete-log* assumption
  - Hash function modelled as a *random oracle* (RO)
  - Argued using (random) oracle replay attacks

Schnorr Signature: Construction

The Setting:

1. We work in group $\mathbb{G} = \langle g \rangle$ of prime order p.
2. A hash function $\mathsf{H} : \{0,1\}^* \mapsto \mathbb{Z}_p$ is used.

Key Generation:

1. Select $z \xleftarrow{\cup} \mathbb{Z}_p$ as the `sk`
2. Set $Z := g^z$ as the `pk`

Signing:

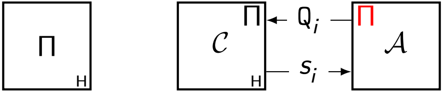
1. Select $r \xleftarrow{\cup} \mathbb{Z}_p$, set $R := g^r$ and $c := \mathsf{H}(m, R)$.
2. The signature on m is $\sigma := (y, R)$ where $y := r + zc$

Verification:

1. Let $\sigma := (y, R)$ and $c := \mathsf{H}(m, R)$.
2. σ is valid if $g^y = RZ^c$

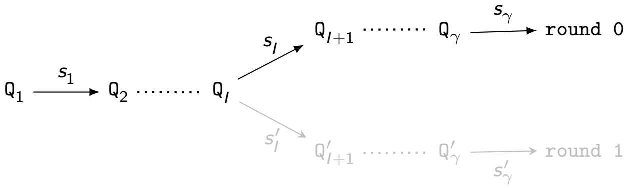
Oracle Replay Attack

- Random oracle H – i^{th} RO query $\mathtt{Q}_i$ replied with s_i



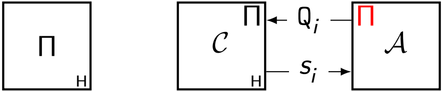
Adversary re-wound to $\mathtt{Q}_I$

Simulation in `round 1` from $\mathtt{Q}_I$ using a *different* random function



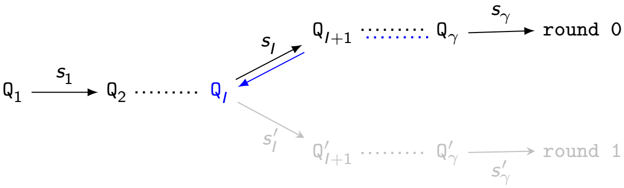
Oracle Replay Attack

- Random oracle H – i^{th} RO query $\mathtt{Q}_i$ replied with s_i.



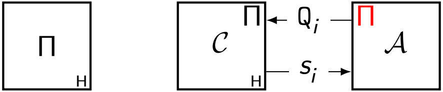
1. Adversary re-wound to $\mathtt{Q}_I$

   Simulation in `round 1` from $\mathtt{Q}_I$ using a *different* random function

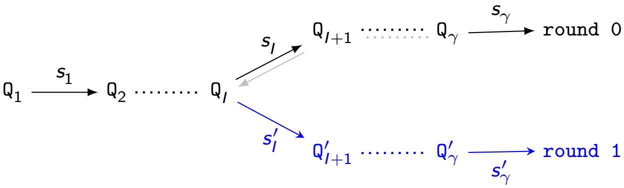


Oracle Replay Attack

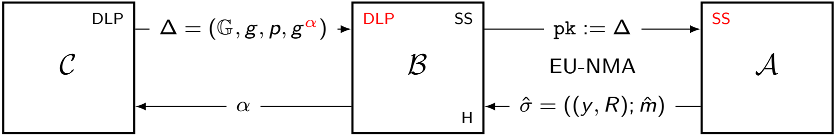
- Random oracle H – i^{th} RO query $\mathtt{Q}_i$ replied with s_i.

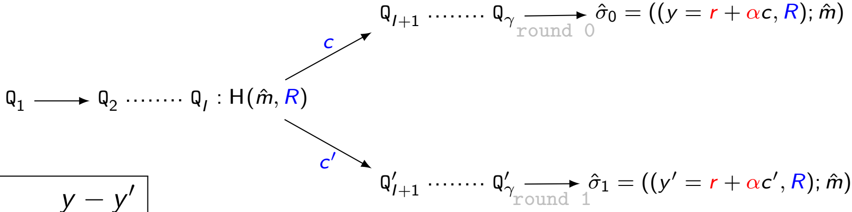


1. Adversary re-wound to $\mathtt{Q}_I$
2. Simulation in `round 1` from $\mathtt{Q}_I$ using a *different* random function



Security of Schnorr Signature, In Brief





$$\alpha = \frac{y - y'}{c - c'}$$

Cost of Oracle Replay Attack

- Forking Lemma [PS00]: bounds success probability of the oracle replay attack (*frk*) in terms of
  1. success probability of the adversary (ϵ)
  2. bound on RO queries (q)

$$\text{DLP} \leq_{O(q/\epsilon^2)} \text{Schnorr Signature}$$

- Analysis done using the Splitting Lemma

[PS00] Pointcheval and Stern. Security arguments for digital signatures and blind signatures. *JoC*, 13

[Seu12] Seurin. On the exact security of Schnorr-type signatures in the random oracle model. *Eurocrypt'12*

Cost of Oracle Replay Attack

- Forking Lemma [PS00]: bounds success probability of the oracle replay attack (*frk*) in terms of
  1. success probability of the adversary (ϵ)
  2. bound on RO queries (q)

$$\text{DLP} \leq_{O(q/\epsilon^2)} \text{Schnorr Signature}$$

- Analysis done using the Splitting Lemma
- The cost: security *degrades* by $O(q)$
  - More or less optimal [Seu12]

[PS00] Pointcheval and Stern. Security arguments for digital signatures and blind signatures. *JoC*, 13

[Seu12] Seurin. On the exact security of Schnorr-type signatures in the random oracle model. *Eurocrypt'12*

General-Forking Lemma

"Forking Lemma is something purely probabilistic, not about signatures" [BN06]

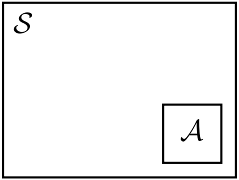
- Abstract version of the Forking Lemma
- Separates out details of simulation (of adversary) from analysis
- A wrapper algorithm used as *intermediary*
  1. Simulate protocol environment to $\mathcal{A}$
  2. Simulate RO as specified by $\mathcal{S}$

[BN06] Bellare and Neven. Multi-signatures in plain public-key model and a general forking lemma. *CCS'06*

General-Forking Lemma

"Forking Lemma is something purely probabilistic, not about signatures" [BN06]

- Abstract version of the Forking Lemma
- Separates out details of simulation (of adversary) from analysis
- A wrapper algorithm used as *intermediary*
  1. Simulate protocol environment to $\mathcal{A}$
  2. Simulate RO as specified by $\mathcal{S}$

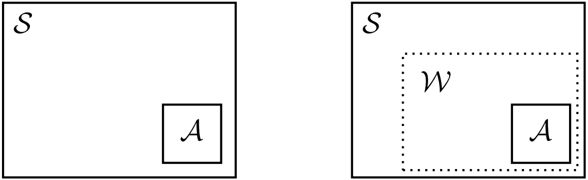


- Structure of a wrapper call: $(I, \sigma) \leftarrow \mathcal{W}(x, s_1, \ldots, s_q; \rho)$

General-Forking Lemma

"Forking Lemma is something purely probabilistic, not about signatures" [BN06]

- Abstract version of the Forking Lemma
- Separates out details of simulation (of adversary) from analysis
- A wrapper algorithm used as *intermediary*
  1. Simulate protocol environment to $\mathcal{A}$
  2. Simulate RO as specified by $\mathcal{S}$



- Structure of a wrapper call: $(I, \sigma) \leftarrow \mathcal{W}(x, s_1, \ldots, s_q; \rho)$

[BN06] Bellare and Neven. Multi-signatures in plain public-key model and a general forking lemma. *CCS'06*

General-Forking Lemma...

General-Forking Algorithm $\mathcal{F}_{\mathcal{W}}(x)$

Pick coins ρ for $\mathcal{W}$ at random

$\{s_1, \ldots, s_q\} \xleftarrow{U} \mathbb{S}$; $(I, \sigma) \leftarrow \mathcal{W}(x, s_1, \ldots, s_q; \rho)$ //round 0
if $(I = 0)$ **then return** $(0, \perp, \perp)$

$\{s_I, \ldots, s'_q\} \xleftarrow{U} \mathbb{S}$; $(I', \sigma') \leftarrow \mathcal{W}(x, s_1, \ldots, s_{I-1}, s'_I, \ldots, s'_q; \rho)$ //round 1
if $(I' = I \wedge s'_I \neq s_I)$ **then return** $(1, \sigma, \sigma')$
else return $(0, \perp, \perp)$

General-Forking Lemma...

General-Forking Algorithm $\mathcal{F}_{\mathcal{W}}(x)$

Pick coins ρ for $\mathcal{W}$ at random

$\{s_1, \ldots, s_q\} \xleftarrow{U} \mathbb{S}$; $(I, \sigma) \leftarrow \mathcal{W}(x, s_1, \ldots, s_q; \rho)$ //round 0

if $(I = 0)$ **then return** $(0, \perp, \perp)$

$\{s_I, \ldots, s'_q\} \xleftarrow{U} \mathbb{S}$; $(I', \sigma') \leftarrow \mathcal{W}(x, s_1, \ldots, s_{I-1}, s'_I, \ldots, s'_q; \rho)$ //round 1

if $(I' = I \wedge s'_I \neq s_I)$ **then return** $(1, \sigma, \sigma')$

else return $(0, \perp, \perp)$

General-Forking Lemma: bounds success probability of the oracle replay attack (*frk*) in terms of

1. success probability of $\mathcal{W}$ (*acc*)
2. bound on RO queries (q)

$$frk \geq acc^2/q$$

Contents

Galindo-Garcia IBS: Features

- Derived from Schnorr signature scheme – *nesting* [GG09]
  - Based on the *discrete-log* (DL) assumption
- Efficient, simple and *does not* use pairing
- Uses two hash functions
- Security argued using nested replay attacks

[GG09] Galindo and Garcia. A Schnorr-like lightweight identity-based signature scheme. *Africacrypt'09*

Galindo-Garcia IBS: Construction

Setting:

1. We work in a group $\mathbb{G} = \langle g \rangle$ of prime order p.
2. Two hash functions $\mathsf{H}, \mathsf{G} : \{0,1\}^* \mapsto \mathbb{Z}_p$ are used.

Set-up:

1. Select $z \xleftarrow{\cup} \mathbb{Z}_p$ as the $\mathtt{msk}$; set $Z := g^z$ as the $\mathtt{mpk}$

Key Extraction:

1. Select $r \xleftarrow{\cup} \mathbb{Z}_p$ and set $R := g^r$.
2. Return $\mathtt{usk} := (y, R)$ as the $\mathtt{usk}$, where $y := r + zc$ and $c := \mathsf{H}(\mathtt{id}, R)$.

Signing:

1. Select $a \xleftarrow{\cup} \mathbb{Z}_p$ and set $A := g^a$.
2. Return $\sigma := (b, R, A)$ as the signature, where $b := a + yd$ and $d := \mathsf{G}(\mathtt{id}, m, A)$.

MULTIPLE FORKING

Multiple Forking: Overview

- Introduced by Boldyreva *et al.* [BPW12]
- Motivation:
  - General Forking: elementary replay attack
    - restricted to *one* RO and single replay attack
  - Multiple Forking: nested replay attack
    - two ROs and multiple (n) replay attacks

[BPW12] Boldyreva *et al.*. Secure proxy signature schemes for delegation of signing rights. *JoC*, 25.

[CMW12] Chow *et al.*. Zero-knowledge argument for simultaneous discrete logarithms. *Algorithmica*, 64(2)

Multiple Forking: Overview

- Introduced by Boldyreva *et al.* [BPW12]
- Motivation:
  - General Forking: elementary replay attack
    - restricted to *one* RO and single replay attack
  - Multiple Forking: nested replay attack
    - two ROs and multiple (n) replay attacks

- Used in [BPW12] to argue security of a DL-based proxy SS
- Used further in
  1. Galindo-Garcia IBS
  2. Chow *et al.* Zero-Knowledge Argument [CMW12]

[BPW12] Boldyreva *et al.*. Secure proxy signature schemes for delegation of signing rights. *JoC*, 25.
[CMW12] Chow *et al.*. Zero-knowledge argument for simultaneous discrete logarithms. *Algorithmica*, 64(2)

Multiple-Forking Algorithm

Multiple-Forking Algorithm $\mathcal{M}_{\mathcal{W},3}$

Pick coins ρ for $\mathcal{W}$ at random

$\{s_1^0, \ldots, s_q^0\} \xleftarrow{U} \mathbb{S};$

$(I_0, J_0, \sigma_0) \leftarrow \mathcal{W}(x, s_1^0, \ldots, s_q^0; \rho)$ //round 0

if $((I_0 = 0) \vee (J_0 = 0))$ **then return** $(0, \perp)$

$\{s_{I_0}^1, \ldots, s_q^1\} \xleftarrow{U} \mathbb{S};$

$(I_1, J_1, \sigma_1) \leftarrow \mathcal{W}(x, s_1^0, \ldots, s_0 I_0 - 1, s_{I_0}^1, \ldots, s_q^1; \rho)$ //round 1

if $\left((I_1, J_1) \neq (I_0, J_0) \vee (s_{I_0}^1 = s_{I_0}^0)\right)$ **then return** $(0, \perp)$

$\{s_{J_0}^2, \ldots, s_q^2\} \xleftarrow{U} \mathbb{S};$

$(I_2, J_2, \sigma_2) \leftarrow \mathcal{W}(x, s_1^0, \ldots, s_0 J_0 - 1, s_{J_0}^2, \ldots, s_q^2; \rho)$ //round 2

if $\left((I_2, J_2) \neq (I_0, J_0) \vee (s_{J_0}^2 = s_{J_0}^1)\right)$ **then return** $(0, \perp)$

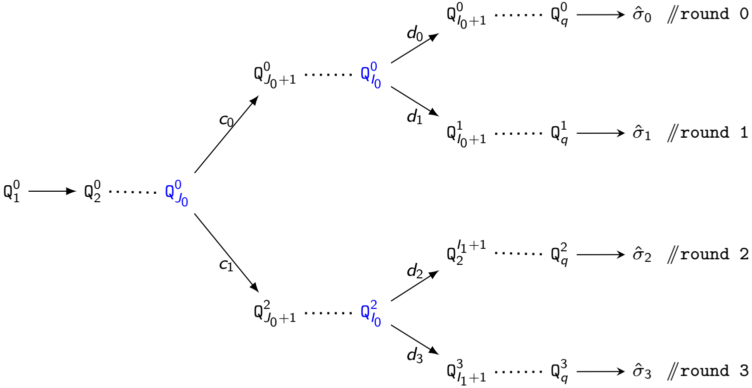
$\{s_3 I_2, \ldots, s_3 q\} \xleftarrow{U} \mathbb{S};$

$(I_3, J_3, \sigma_3) \leftarrow \mathcal{W}(x, s_1^0, \ldots, s_0 J_0 - 1, s_{J_0}^2, \ldots, s_{I_2-1}^2, s_3 I_2, \ldots, s_3 q; \rho)$ //round 3

if $((I_3, J_3) \neq (I_0, J_0) \vee (s_3 I_0 = s_2 I_0))$ **then return** $(0, \perp)$

return $(1, \{\sigma_0, \ldots, \sigma_3\})$

...Multiple-Forking Algorithm...



Multiple-Forking Lemma

Multiple-Forking Lemma: bounds success probability of nested replay attack (*mfrk*) in terms of

1. success probability of $\mathcal{W}$ (*acc*)
2. bound on RO queries (q)
3. number of rounds of forking (n)

$$mfrk \geq acc^{n+1}/q^{2n}$$

Multiple-Forking Lemma

Multiple-Forking Lemma: bounds success probability of nested replay attack (*mfrk*) in terms of

1. success probability of $\mathcal{W}$ (*acc*)
2. bound on RO queries (q)
3. number of rounds of forking (n)

$$mfrk \geq acc^{n+1}/q^{2n}$$

Follows from condition $\mathsf{F} : (I_n, J_n) = (I_{n-1}, J_{n-1}) = \ldots = (I_0, J_0)$

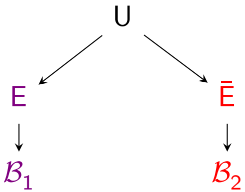
Degradation: $O\left(q^{2n}\right)$

- Cost per forking (involving two ROs): $O\left(q^2\right)$

SECURITY ARGUMENT

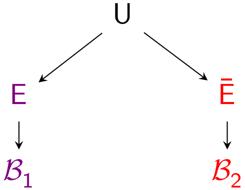
Original Security Argument

- Two reductions: $\mathcal{B}_1$ and $\mathcal{B}_2$ depending on the type of adversary (event E and $\bar{\text{E}}$)
    - DLP $\leq$ GG-IBS



Original Security Argument

- Two reductions: $\mathcal{B}_1$ and $\mathcal{B}_2$ depending on the type of adversary (event E and $\bar{\text{E}}$)
  - DLP $\leq$ GG-IBS



Reduction	Success Prob. ($\approx$)	Forking Algorithm
$\mathcal{B}_1$	$\epsilon^2/q_{\mathsf{G}}^3$	General Forking ($\mathcal{F}_{\mathcal{W}}$)
$\mathcal{B}_2$	$\epsilon^4/(q_{\mathsf{H}}q_{\mathsf{G}})^6$	Multiple Forking ($\mathcal{M}_{\mathcal{W},3}$)

Original Security Argument: Flaws

- We found several problems with $\mathcal{B}_1$ and $\mathcal{B}_2$
  1. $\mathcal{B}_1$: Fails in the standard security model for IBS
  2. $\mathcal{B}_2$: All the adversarial strategies were not covered
- Simulation is distinguishable from real execution!

[CKK12] Chatterjee *et al.*. Galindo-Garcia identity-based signature, revisited. *ICISC'12*

Original Security Argument: Flaws

- We found several problems with $\mathcal{B}_1$ and $\mathcal{B}_2$
  1. $\mathcal{B}_1$: Fails in the standard security model for IBS
  2. $\mathcal{B}_2$: All the adversarial strategies were not covered
- Simulation is distinguishable from real execution!

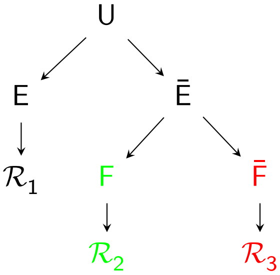
- Contribution: *fixed* the security argument
  - Slightly tighter reduction [CKK12]

[CKK12] Chatterjee *et al.*. Galindo-Garcia identity-based signature, revisited. *ICISC'12*

Fixed Security Argument

- Type $\bar{\mathsf{E}}$ further split: type F and $\bar{\mathsf{F}}$

  F: $\mathcal{A}$ makes target $\mathsf{G}(\cdot,\cdot,\cdot)$ before target $\mathsf{H}(\cdot,\cdot)$ ($\mathsf{G} < \mathsf{H}$)

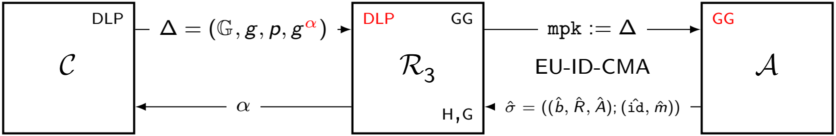


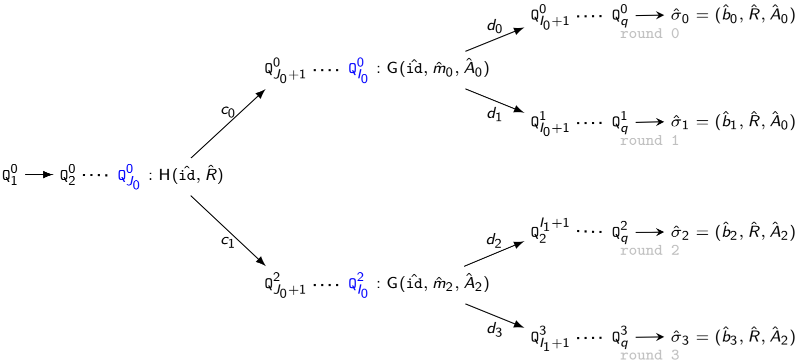
1. $\mathcal{R}_1$ *addresses* problems with $\mathcal{B}_1$ + Coron's Technique
2. $\mathcal{R}_2$ *covers* unaddressed adversarial strategy in $\mathcal{B}_2$ (*i.e.*, $\mathsf{H} < \mathsf{G}$)
3. $\mathcal{R}_3$ *same* as the original reduction $\mathcal{B}_2$

Fixed Security Argument

Reduction	Success Prob. ($\approx$)	Forking Used
$\mathcal{R}_1$	$\frac{\epsilon^2}{q_G q_\varepsilon}$	$\mathcal{F}_{\mathcal{W}}$
$\mathcal{R}_2$	$\frac{\epsilon^2}{(q_H+q_G)^2}$	$\mathcal{M}_{\mathcal{W},1}$
$\mathcal{R}_3$	$\frac{\epsilon^4}{(q_H+q_G)^6}$	$\mathcal{M}_{\mathcal{W},3}$

Reduction $\mathcal{R}_3$





Degradation

- Degradation: $O\left(q^6\right)$
  - Reason: cost per forking is $O\left(q^2\right)$

Degradation

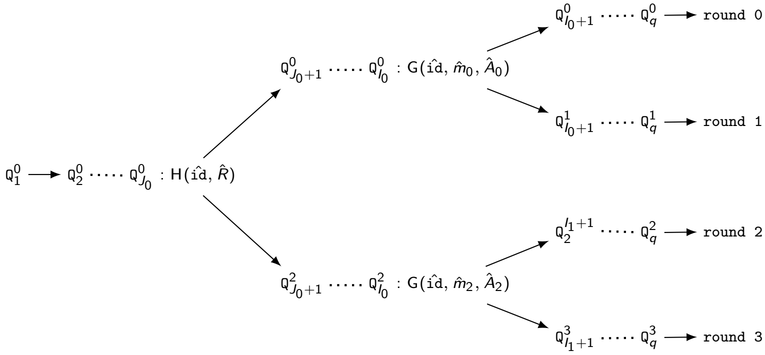
- Degradation: $O\left(q^6\right)$
  - Reason: cost per forking is $O\left(q^2\right)$

- Can we improve?

Contents

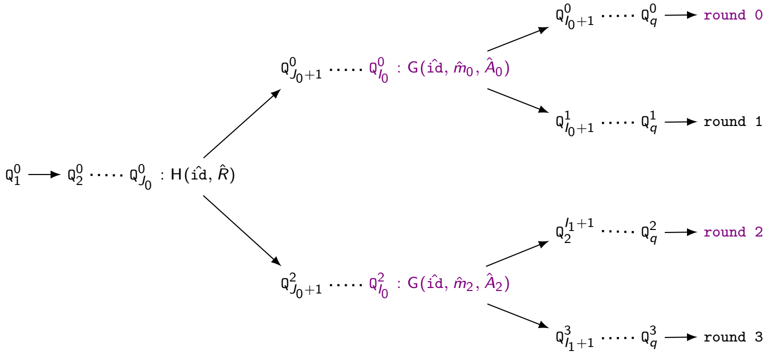
The Intuition

- Recall, condition F : $(I_3, J_3) = (I_2, J_2) = (I_1, J_1) = (I_0, J_0)$



The Intuition

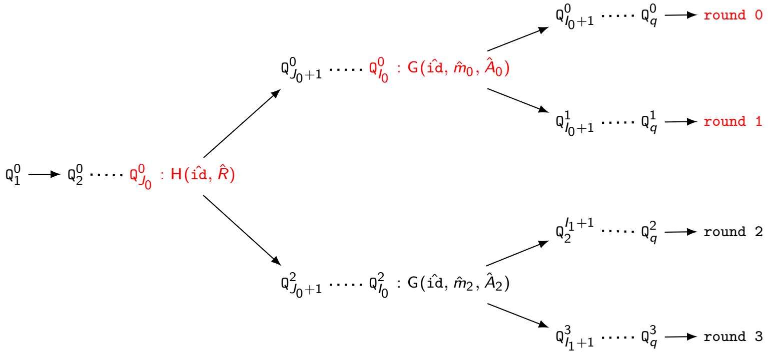
- Recall, condition F : $(I_3, J_3) = (I_2, J_2) = (I_1, J_1) = (I_0, J_0)$



- Observations:
  1. *Independence* condition O_1: I_2 *need not* equal I_0

The Intuition

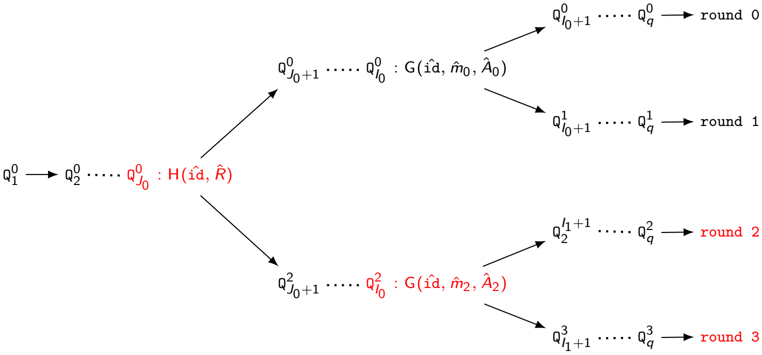
- Recall, condition F : $(I_3, J_3) = (I_2, J_2) = (I_1, J_1) = (I_0, J_0)$



- Observations:
  1. *Independence* condition O_1: I_2 *need not* equal I_0
  2. *Dependence* condition O_2: $(I_1 = I_0)$ can *imply* $(J_1 = J_0)$

The Intuition

- Recall, condition F : $(I_3, J_3) = (I_2, J_2) = (I_1, J_1) = (I_0, J_0)$



- Observations:
  1. *Independence* condition O_1: I_2 *need not* equal I_0
  2. *Dependence* condition O_2: $(I_1 = I_0)$ can *imply* $(J_1 = J_0)$ (similarly $(I_3 = I_2)$ can *imply* $(J_3 = J_2)$)

The Intuition...

Effect of O_1 and O_2 on F : $(I_3, J_3) = (I_2, J_2) = (I_1, J_1) = (I_0, J_0)$

- O_1: I_2 *need not* equal I_0

$$(I_3, J_3) = (I_2, J_2) \wedge (J_2 = J_0) \wedge (I_1, J_1) = (I_0, J_0)$$

- O_2: $(I_1 = I_0) \implies (J_1 = J_0)$ and $(I_3 = I_2) \implies (J_3 = J_2)$

$$(I_3 = I_2 = I_1 = I_0) \wedge (J_2 = J_0)$$

...The Intuition...

Effect of O_1 and O_2 on F : $(I_3, J_3) = (I_2, J_2) = (I_1, J_1) = (I_0, J_0)$

- O_1: I_2 *need not* equal I_0

$$(I_3, J_3) = (I_2, J_2) \wedge (J_2 = J_0) \wedge (I_1, J_1) = (I_0, J_0)$$

- O_2: $(I_1 = I_0) \implies (J_1 = J_0)$ and $(I_3 = I_2) \implies (J_3 = J_2)$

$$(I_3 = I_2 = I_1 = I_0) \wedge (J_2 = J_0)$$

- Together, O_1 & O_2:

$$(I_3 = I_2) \wedge (I_1 = I_0) \wedge (J_2 = J_0)$$

...The Intuition...

Effect of O_1 and O_2 on F : $(I_3, J_3) = (I_2, J_2) = (I_1, J_1) = (I_0, J_0)$

- O_1: I_2 *need not* equal I_0

$$(I_3, J_3) = (I_2, J_2) \wedge (J_2 = J_0) \wedge (I_1, J_1) = (I_0, J_0)$$

- O_2: $(I_1 = I_0) \implies (J_1 = J_0)$ and $(I_3 = I_2) \implies (J_3 = J_2)$

$$(I_3 = I_2 = I_1 = I_0) \wedge (J_2 = J_0)$$

- Together, O_1 & O_2:

$$(I_3 = I_2) \wedge (I_1 = I_0) \wedge (J_2 = J_0)$$

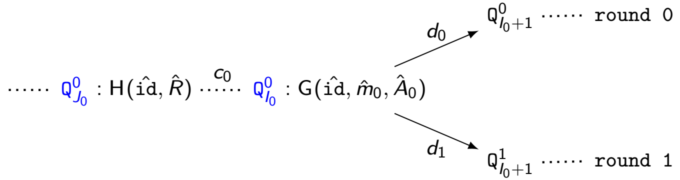
Intuitively, degradation reduced to $O(q^3)$

- In general, degradation reduced to $O(q^n)$

MORE ON (IN)DEPENDENCE

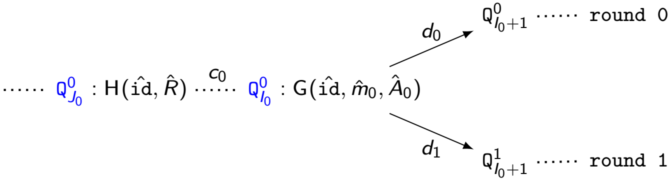
Inducing RO Dependence

- Consider `round 0` and `round 1` of simulation for GG-IBS



Inducing RO Dependence

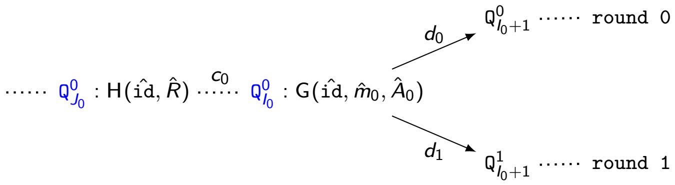
- Consider `round 0` and `round 1` of simulation for GG-IBS



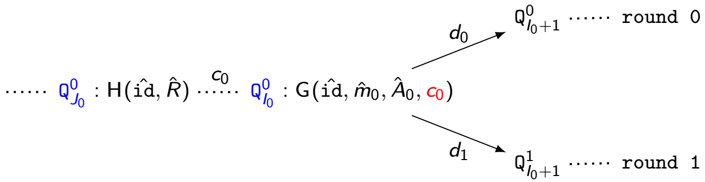
- Need to explicitly ensure that $(J_1 = J_0)$

Inducing RO Dependence

- Consider `round 0` and `round 1` of simulation for GG-IBS



- Need to explicitly ensure that $(J_1 = J_0)$



Inducing RO Dependence

- Consider round 0 and round 1 of simulation for GG-IBS

$$\cdots\cdots\ \mathtt{Q}^0_{J_0} : \mathsf{H}(\hat{\mathtt{id}}, \hat{R}) \overset{c_0}{\cdots\cdots}\ \mathtt{Q}^0_{I_0} : \mathsf{G}(\hat{\mathtt{id}}, \hat{m}_0, \hat{A}_0)$$

$$\xrightarrow{d_0} \mathtt{Q}^0_{I_0+1} \cdots\cdots \texttt{round 0}$$

$$\xrightarrow{d_1} \mathtt{Q}^1_{I_0+1} \cdots\cdots \texttt{round 1}$$

- Need to explicitly ensure that $(J_1 = J_0)$

$$\cdots\cdots\ \mathtt{Q}^0_{J_0} : \mathsf{H}(\hat{\mathtt{id}}, \hat{R}) \overset{c_0}{\cdots\cdots}\ \mathtt{Q}^0_{I_0} : \mathsf{G}(\hat{\mathtt{id}}, \hat{m}_0, \hat{A}_0, c_0)$$

$$\xrightarrow{d_0} \mathtt{Q}^0_{I_0+1} \cdots\cdots \texttt{round 0}$$

$$\xrightarrow{d_1} \mathtt{Q}^1_{I_0+1} \cdots\cdots \texttt{round 1}$$

- Hence, $(I_1 = I_0) \implies (J_1 = J_0)$!

...Inducing RO Dependence...

Definition (RO Dependence)

An RO H_2 is η-dependent on RO H_1 ($H_1 \prec H_2$) if:

1. $(1 \leq J < I \leq q)$ and
2. $\Pr[(J' \neq J) \mid (I' = I)] \leq \eta$

...Inducing RO Dependence...

Definition (RO Dependence)

An RO H_2 is η-dependent on RO H_1 ($\mathsf{H}_1 \prec \mathsf{H}_2$) if:

1. $(1 \leq J < I \leq q)$ and
2. $\Pr[(J' \neq J) \mid (I' = I)] \leq \eta$

Claim (Binding induces dependence)

Binding H_2 to H_1 *induces* a RO dependence $\mathsf{H}_1 \prec \mathsf{H}_2$ with $\eta_b := q_1(q_1 - 1)/|\mathbb{R}_1|$.

- q_1: upper bound on queries to H_1
- $\mathbb{R}_1$: range of H_1

Galindo-Garcia IBS with Binding

Setting:

1. We work in a group $\mathbb{G} = \langle g \rangle$ of prime order p.
2. Two hash functions $\mathsf{H}, \mathsf{G} : \{0,1\}^* \mapsto \mathbb{Z}_p$ are used.

Set-up:

1. Select $z \stackrel{\mathrm{U}}{\leftarrow} \mathbb{Z}_p$ as the msk; set $Z := g^z$ as the mpk

Key Extraction:

1. Select $r \stackrel{\mathrm{U}}{\leftarrow} \mathbb{Z}_p$ and set $R := g^r$.
2. Return $\mathtt{usk} := (y, R)$ as the usk, where $y := r + zc$ and $c := \mathsf{H}(\mathtt{id}, R)$.

Signing:

1. Select $a \stackrel{\mathrm{U}}{\leftarrow} \mathbb{Z}_p$ and set $A := g^a$.
2. Return $\sigma := (b, R, A)$ as the signature, where $b := a + yd$ and $d := \mathsf{G}(m, A, c)$.

Effects of (In)Dependence

- Enables better (but involved) analysis
  - Imparts a structure to underlying set of random tapes
  - Analysis using the Splitting Lemma (twice) in place of an Extended Splitting Lemma

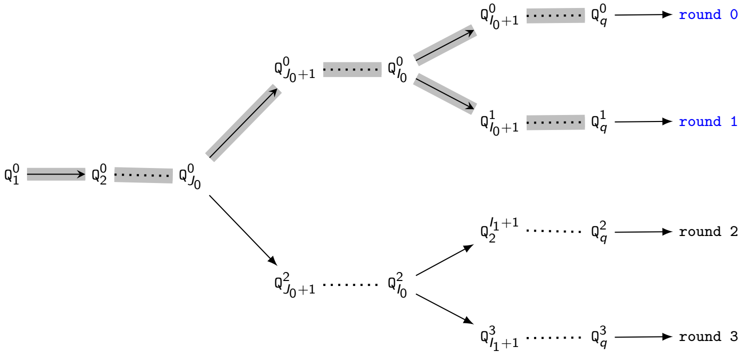
Effects of (In)Dependence

- Enables better (but involved) analysis
  - Imparts a structure to underlying set of random tapes
  - Analysis using the Splitting Lemma (twice) in place of an Extended Splitting Lemma
- Effective degradation for GG-IBS: $\mathrm{O}\left(q^3\right)$
  - Cost per forking (involving two ROs): $\mathrm{O}\left(q\right)$

The Conceptual Wrapper

- Observations *better* formulated using a conceptual wrapper
  - Clubs two (consecutive) executions of the original wrapper
  - Denoted by $\mathcal{Z}$

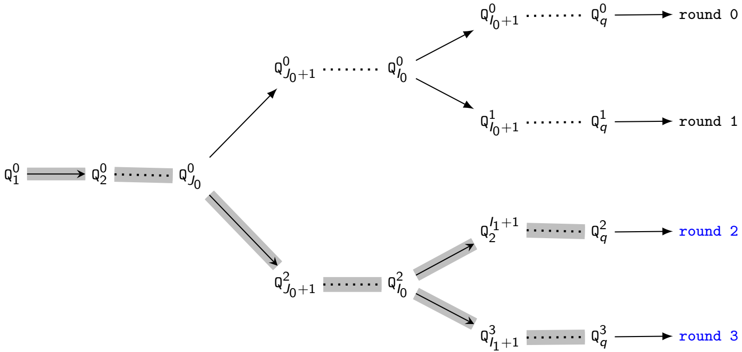
$$(I_k, J_k, \sigma_k), (I_{k+1}, J_{k+1}, \sigma_{k+1})) \leftarrow \mathcal{Z}\left(x, \mathtt{S}^k, \mathtt{S}^{k+1}; \rho\right)$$



The Conceptual Wrapper

- Observations *better* formulated using a conceptual wrapper
  - Clubs two (consecutive) executions of the original wrapper
  - Denoted by $\mathcal{Z}$

$$(I_k, J_k, \sigma_k), (I_{k+1}, J_{k+1}, \sigma_{k+1})) \leftarrow \mathcal{Z}\left(x, \mathtt{S}^k, \mathtt{S}^{k+1}; \rho\right)$$



Abstracting (In)Dependence

- Index Dependence: It is possible to design protocols such that, for the k^{th} invocation of $\mathcal{Z}$, $(I_{k+1} = I_k) \implies (J_{k+1} = J_k)$.
- Index Independence: It is not necessary for the I indices across $\mathcal{Z}$ to be the same
  - I_k need not be equal to $I_{k-2}, I_{k-4}, \ldots, I_0$ for $k = 2, 4, \ldots, n-1$

[CK13a] Chatterjee and Kamath. A Closer Look at Multiple Forking: Leveraging (In)dependence for a Tighter Bound – *IACR eprint archive*, 2013/651

Abstracting (In)Dependence

- Index Dependence: It is possible to design protocols such that, for the k^{th} invocation of $\mathcal{Z}$, $(I_{k+1} = I_k) \implies (J_{k+1} = J_k)$.
- Index Independence: It is not necessary for the I indices across $\mathcal{Z}$ to be the same
  - I_k need not be equal to $I_{k-2}, I_{k-4}, \ldots, I_0$ for $k = 2, 4, \ldots, n-1$

- We formulated a unified model for multiple forking [CK13a]
  - Four different cases depending on applicability of O_1 & O_2

[CK13a] Chatterjee and Kamath. A Closer Look at Multiple Forking: Leveraging (In)dependence for a Tighter Bound – *IACR eprint archive*, 2013/651

Contents

Construction of IBS from sID-IBS

- sID Model: a weaker model
  - Adversary has to, beforehand, commit to the *target* identity
- Goal: construct ID-secure IBS from sID-secure IBS
  1. without random oracles
  2. with sub-exponential degradation
- Tools used:
  1. Chameleon Hash Function (CHF)
  2. GCMA-secure PKS

[CK13b] Chatterjee and Kamath. From selective-id to full-id IBS without random oracles. *SPACE'13*

Construction of IBS from sID-IBS

- sID Model: a weaker model
  - Adversary has to, beforehand, commit to the *target* identity
- Goal: construct ID-secure IBS from sID-secure IBS
  1. without random oracles
  2. with sub-exponential degradation
- Tools used:
  1. Chameleon Hash Function (CHF)
  2. GCMA-secure PKS

- Main result: EU-ID-CMA-IBS $\equiv$ (EU-sID-CMA-IBS)+(EU-GCMA-PKS)+(CR-CHF)
- Further: EU-ID-CMA-IBS $\equiv$ (EU-wID-CMA-IBS)+(EU-GCMA-PKS)+(CR-CHF)

[CK13b] Chatterjee and Kamath. From selective-id to full-id IBS without random oracles. *SPACE'13*

Contents

Conclusion and Future Work

Conclusions:

- Identified flaws in security argument of GG-IBS
- Came up with a tighter security bound for GG-IBS
- Constructed IBS from weaker IBS

Future directions:

- Is the bound optimal?
- Other applications for RO dependence?
  - Γ-protocols [YZ13]
  - Extended Forking Lemma [YADV+12]
- Other techniques to induce RO dependence

[YZ13] Yao and Zhao. Online/offline signatures for low-power devices. *IEEE IFS*, 8(2)

[YADV+12] Yousfi-Alaoui *et al.*. Extended Security Arguments for Signature Schemes. *Africacrypt'12*

THANK YOU!